Rethinking Virtual Link Mapping in Network Virtualization

Khoa TD 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*—Virtual Network Embedding (VNE) that addresses the embedding problems of heterogeneous virtual networks onto a physical limited-capacity infrastructure efficiently is a major challenge in network virtualization (NV). VNE is computationally intractable when considering various constraints on nodes and links, and is also known as $\mathcal{NP}$-hard even in offline embedding. Although the VNE problems have received much attentions over recent decades with a vast number of VNE solutions, the majority of them only focus on VNE node mapping, whilst leaving the link mapping stage for the shortest path method or multicommodity flow (MCF) algorithm. We persuasively argue that node and link mappings equally play pivotal roles to approach an efficient VNE solution. In this paper, we reassess the role of link mapping stage in VNE problem, and then propose a novel intelligent VNE orchestration which effectively implements a distributed parallel model to reduce the operation time remarkably. Extensive evaluation results show that our proposed algorithm is not only faster than state-of-the-art VNE algorithms in speed, but also better in all performance metrics.**

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

I. Introduction

NV is considered as a promising paradigm to pave the success of the future generation networks such as 5G-and-beyond [1], virtualised IoT networks [2]. NV allows to share the physical network resources among multiple virtual network requests (VNRs), enabling an isolated coexistence of multiple virtual networks (VNs) on a single substrate network (SN). This key technology brings more efficient resource utilization to the SN and offer a great opportunity of implementing as well as evaluating new network protocols or architecture designs. It also prevents an unnecessary expansion of network infrastructure. The critical challenge of embedding VNRs with diverse topology and stringent resources onto the underlying physical network is known as Virtual Network Embedding (VNE) problem.

In essence, VNE process that enables mapping requested VNs onto the underlying shared physical network can be decoupled 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]. However, it is important to note that VLiM problem is de facto more challenging than the analogue VNoM due to the requirements that all the substrate links along which a virtual link is mapped onto must have enough residual capacities to support the bandwidth requirement of the virtual link, which results in the bandwidth fragmentation problem more likely to occur. In practise, the most common failures of mapping VNRs invariably emanate from the ineffective link mapping algorithm [4]. Consequently, we presume that an appropriate design of VLiM mechanism will not only improve the efficiency of link resource utilization, but also increase the number of accepted VNRs.

In fact, the formulated optimization models such as Integer Linear Programming (ILP) are commonly proposed to achieve optimal VNE solutions, but they cannot be actually tailored for online VNE problems due to their intricacy, non-scalability and non-polynomial time issue. Instead of exact methods, most of research papers have simply adopted light-weighted heuristic algorithms to tackle the aforementioned impediments of the formulated optimization models. However, most research work [3]–[10] engrosses in seeking an efficient node mapping, but it seems to underestimate the link mapping stage since they completely entrust to k-shortest path or multicommodity flow (MCF) algorithms, which definitely restricts VNE link mapping options.

Towards 5G-and-beyond networking and smart IoT, VNE problem where the physical infrastructure allows splittable and unsplittable resource configurations is indisputably an essential research topic in Software Defined Network (SDN), Network Function Virtulization (NFV) and Future Edge Clouds. Splittability permits a virtual link demand to be embedded on multiple substrate paths whilst if it is mapped onto a single physical path with fixed node mapping, which reduces to unsplittable configuration. Although splittable-based embedding is literally expected to obtain better resource utilization, it may generate a larger overhead to consistently maintain the network state [11]. Due to the aforementioned reasons, we merely focused on unsplittable mapping in this paper. In contrast, parallel algorithms can be tremendously beneficial in dealing with the intricate computing tasks thanks to lower hardware costs for computing recently.

In this paper, we propose a novel GA-based VNE algorithm that is relied on new design of the crossover operator for VLiM. The multi-constrained fitness function considering the embedding cost, hop-count and propagation delay has driven the proposal to an efficient 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. To the best of our knowledge, this is the first paper that applies an elastic crossover mechanism in GA algorithm to VNE problems. This paper is an extension of [12] towards the crossover operator and the improved fitness function, which not only achieves better performance than [12], but also outperforms state-of-the-art VNE algorithms.

The remainder of this paper is organized as follows: the network model is formulated in Section II and then we present our proposed distributed parallel GA-based algorithm for VNE link mapping in Section III. The performance evaluation is introduced in Section IV whilst the related work is presented in Section V. Section VI is finally a conclusion of this paper.

II. Network Model and Problem Descriptions

A. Virtual Network Assignment

The VNE substrate network is modelled as a weighted undirected graph $G^s = (N^s, L^s)$, in which N^s is the set of all substrate nodes and L^s is the set of all substrate links. Basically, each substrate node $n^s \in N^s$ that has a geographic location $loc(n^s)$ is characterised by the available CPU capacity $c\,(n^s)$, whereas each substrate link $l^s \in L^s$ between any two substrate nodes has a finite bandwidth capacity $b\,(l^s)$. Memory and storage resources will be omitted in this article for simplification. In VNE research, we can model the i^{th} arriving VNR as a weighted undirected graph denoted as $G_i^v = (N_i^v, L_i^v)$, where N_i^v is the set of all virtual nodes and L_i^v is the set of all virtual links towards the i^{th} VNR. Each virtual node $n_i^v \in N_i^v$ is inherently characterised by a requested CPU capacity $c(n_i^v)$, whilst 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 has a requested bandwidth capacity $b\,(l_i^v)$. Each VNR has a preferable mapping radius $D(n_i^v)$ that discloses how far a virtual node n_i^v can be placed from its location identifier $loc(n_i^v)$. Mapping the i^{th} VNR G_i^v onto the SN G^s can be decomposed into two main components as determined above: VNoM and VLiM. Under node mapping stage, a virtual node from 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 that is mapped on the substrate node n^s, and the distance between the geographical locations of node i^s and j^d is measured by $\mathcal{D}(i^s, j^d)$. Besides $R_N(n^s)$ denotes the residual/available CPU capacity of a substrate node. In fact, a virtual link is mostly embedded on the corresponding substrate path with one or more substrate links. As such, this unsplittable link embedding can be denoted by $\mathcal{A}_{\mathcal{L}} : L_i^v \rightarrow L^s$ whilst $l_i^v = (s_i^v, d_i^v) \in L_i^v$, $\mathcal{E}^s(\mathcal{A}_{\mathcal{L}}(l_i^v))$ is a set of all possible substrate paths from source node $\mathcal{A}_{\mathcal{N}}(s_i^v)$ 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

From the InPs' perspective, the main VNE objective is to maximize their accumulated revenues while keeping its embedding cost minimal. In this paper, the generated revenue of InPs is practically calculated as the sum of total virtual resources embedded on the SN over time. Accordingly, the revenue of i^{th} VNR G_i^v is computed as below:

$$\mathcal{R}(G_i^v) = w_b * \sum_{l_i^v \in L_i^v} b(l_i^v) + w_n * \sum_{n_i^v \in N_i^v} c(n_i^v) \tag{9}$$

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_b and w_n are the unit weights of the mapped bandwidth and CPU resources respectively.

Cost: we likewise characterize the cost of the i^{th} VNE $C(G_i^v)$ as the sum of total network resources allocated to the i^{th} VN.

$$C(G_i^v) = \sum_{n_i^v \in N_i^v} c(n_i^v) + \sum_{l_i^v \in L_i^v} \sum_{l^s \in L^s} f_{l^s}^{l_i^v} \tag{10}$$

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

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

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:

$$\mathcal{R}_m(L^s) = \sum_{l^s \in L^s} (b(l^s) - \sum_{l_i^v \rightarrow l^s} b(l_i^v)) \tag{12}$$

Meanwhile, there are new VNRs arrived, the InP will intrinsically calculate the residual network resources, and then attempts to embed the corresponding VNRs onto the physical network depending on such achieved remaining resource information. Higher remaining bandwidth would bring higher chance of accepting the prospective virtual networks.

Fitness Function (FF): the fitness values of each solution determine which one will reproduce and remain "alive" in the next generation, relevant to the predefined objectives to be optimized in our proposed GA-based algorithm in Section III-B. As a result, this function is utilized to examine the quality of each VLiM solution among several feasible ones so that its values can provide a scientific proof for electing the corresponding solutions in GA stages. In details, we take the cost of embedding a VNR into consideration in this paper, so solutions with less cost generated are definitely preferable. Moreover, we consider hop-count as an important factor into FF as it is substantially associated with bandwidth consumption. This means that less hop-count solution would consume less bandwidth, and then leaves more residual network bandwidth, increasing the possibility of the upcoming VNRs being accepted. The propagation delay of VLiM solutions is also estimated and added into FF as another constraint accompanying with the hop-count attribute in order to construct a multi-constrained fitness function. Fitness function $\mathcal{F}(\mathcal{S}_z)$ is eventually calculated as below:

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

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 factors.

III. Distributed Parallel Resource-Allocation Algorithm

A. Backgrounds and Ideas

Parallel and distributed computing has recently emerged as an effective mechanism to tackle large and complex problems with less time consuming and lower cost by supporting the concurrency. In addition, GA algorithm is an appealing AI approach for dealing with both constrained and unconstrained optimization problems by adopting the natural selection idea.

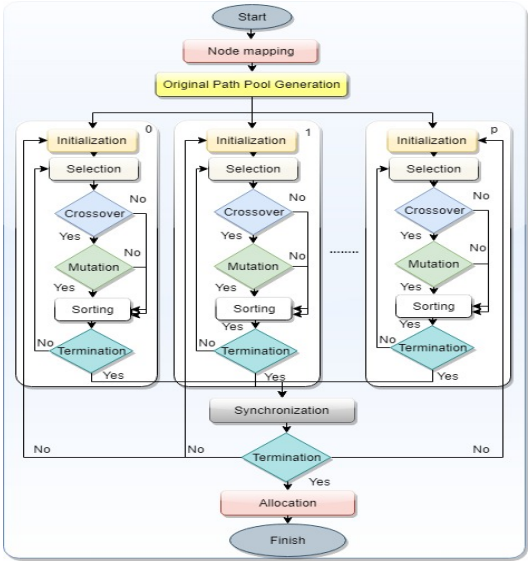


Fig. 1: *Parallel operation scheme*

A typical GA consists of four primary operators: initialization, selection, crossover and mutation. Crossover operator can be recognized as an exploitation phase in which the global optimum is positively expected to discover, so we argue that a creative crossover operator would improved the efficiency of our GA algorithm driven by an appropriate FF. Inspired from the DNA replication process in [13], this mechanism allows cell division to occur. A parental cell can be

split into two or more daughter cells by this process. They can inherit diverse combinations of partial or all DNA. In this research work, we redesign a novel crossover operator for the proposed GA algorithm. This new mechanism allows to proportionally exchange the random genes between the parental chromosomes to generate new offsprings.

As discussed, we assume that the role of link mapping stage in VNE is being underestimated, and the fastest speed of shortest path method can be vanquished by implementing an ingenious parallel operation scheme. Thus, we propose a novel intelligent GA-based orchestration algorithm for VLiM stage, running on a predefined number of independently distributed parallel machines (e.g. virtual machines) to generate the feasible solutions denoted as chromosomes. To prove our hypothesis, we deploy a simple Greedy node mapping as the same with G-SP algorithm in [14] due to its simplicity. This selection can not only guarantee the rapid embedding speed, but also maximize the residual network resources leading more successfully allocated node mapping requests in future. Our proposed parallel GA scheme is presented in Fig 1.

B. Distributed Parallel Genetic Algorithm (DPGA)

As depicted in Fig 1, we present the functioning procedures sequentially working under a single master node such as node mapping, original path pool generation, synchronization, allocation, whereas the others handle the parallel GA algorithms to find the feasible solutions for VNE link mapping working as several slave nodes. Each slave machine is independently running with defined iterations, and then the best-matching feasible VLiM outcome is selected amongst the parallel machines. Unlike other research papers embedding virtual link requests sequentially, our proposed algorithm enables to map all link requests of a VNR altogether. A chromosome $\mathcal{C}_f$ including several genes g_i^j denotes a feasible link embedding solution for all virtual link requests of a VNR. Each gene g_i^j, where i and j indicate its current chromosome and virtual link respectively, is associated with a substrate path that is a feasible solution for a virtual link request. It means that the number of genes corresponding to a VNR constitute a chromosome.

1) Initial path pool generation

We deliberately create the potential path database for the requested virtual links before conducting link mapping procedures. For each pair of source-destination, a k-shortest path algorithm e.g. Dijkstra's algorithm is simply deployed to identify k-shortest paths during the path pool generation. This intrinsic process can be obviously determined prior to the arrival of online VNRs since the substrate network is static.

2) Slave node

Population Initialization: each slave machine usually begins to handle the proposed GA algorithm with a population initialization step where each chromosome $\mathcal{C}_f$ defines a feasible solution. It is assumed that there are M chromosomes and each chromosome has N genes. An initial population $\mathcal{P}$ (MxN size) at the k^{th} machine can be represented as below:

$$\mathcal{P} = \begin{bmatrix} \mathcal{C}_1 \\ \mathcal{C}_2 \\ \vdots \\ \mathcal{C}_f \\ \vdots \\ \mathcal{C}_M \end{bmatrix} = \begin{bmatrix} g_1^1 & \cdots & g_1^j & \cdots & g_1^N \\ g_2^1 & \cdots & g_2^j & \cdots & g_2^N \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ g_f^1 & \cdots & g_f^j & \cdots & g_f^N \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ g_M^1 & \cdots & g_M^j & \cdots & g_M^N \end{bmatrix} \quad (14)$$

To construct a chromosome, each gene associated with the potential embedding solution of a virtual link request shall be uniformly selected from the initial path pool in random. This solution must pass the feasibility-checking process to become a potential link solution. This checking procedure is actually necessary since it guarantees the underlying SN still has adequate residual resources to embed the corresponding request. All N potential genes that have already passed the feasibility check will compose a chromosome, considering as a feasible solution for the VLiM.

Selection: in order to enhance the parallelism degree, the parents are randomly chosen from the initial population with replacement. We apply the fitness-based proportionate selection scheme to adopt parents from the initial population, which conceptually relies on the cumulative sum of the fitness relative weights (13). However, the children produced might either achieve better or worse quality than their parents in next operators due to the randomness selection.

$$\mathcal{P} = \begin{bmatrix} C_1 \\ \vdots \\ C_s \\ \vdots \\ C_r \\ \vdots \\ C_M \\ C_{M+1} \\ C_{M+2} \end{bmatrix} = \begin{bmatrix} g_1^1 & \cdots & \cdots & g_1^{j^{c+2}} & g_1^{j^{c+3}} & \cdots & \cdots & \cdots & g_1^N \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ g_s^1 & \cdots & \cdots & \cdots & g_s^{j^{c+3}} & g_s^{j^{c+4}} & \cdots & \cdots & g_s^N \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ g_r^1 & \cdots & g_r^{j^{c+1}} & g_r^{j^{c+2}} & \cdots & \cdots & g_r^{j^{c+5}} & \cdots & g_r^N \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ g_M^1 & \cdots & \cdots & g_M^{j^{c+2}} & g_M^{j^{c+3}} & \cdots & \cdots & \cdots & g_M^N \\ g_s^1 & \cdots & g_r^{j^{c+1}} & g_r^{j^{c+2}} & \cdots & \cdots & g_r^{j^{c+5}} & \cdots & g_r^N \\ g_r^1 & \cdots & \cdots & \cdots & g_s^{j^{c+3}} & g_s^{j^{c+4}} & \cdots & \cdots & g_s^N \end{bmatrix} \quad (14b)$$

Crossover: this is literally considered as one of the most important step in GA algorithm, which primarily combines the parental chromosomes to produce new offsprings in next generations. We define $\mathcal{C}_s$ and $\mathcal{C}_r$ as the parental chromosomes with the corresponding indices s and r in the initial population. Different from typical GA crossover operator that normally employs one or more static random crossover points to exchange the consecutive sequence of genes between two parental chromosomes, we proceed the crossover phase on selected chromosomes with different sequences of random numbers. Specifically, each related child has a different number of random genes to exchange with the others and vice versa. For example, Fig 2 illustrates our proposed crossover operator where the 1^{st} chromosome exchanges the 1^{st}, 2^{nd} and 5^{th} genes with the other. On the other hand, the 2^{nd} chromosome gets new 3^{rd} and 4^{th} genes from the first chromosome. This novel operation will provide an elastic crossover mechanism producing resilient offsprings, which is particularly desirable to improve the exploitation phase. Moreover,

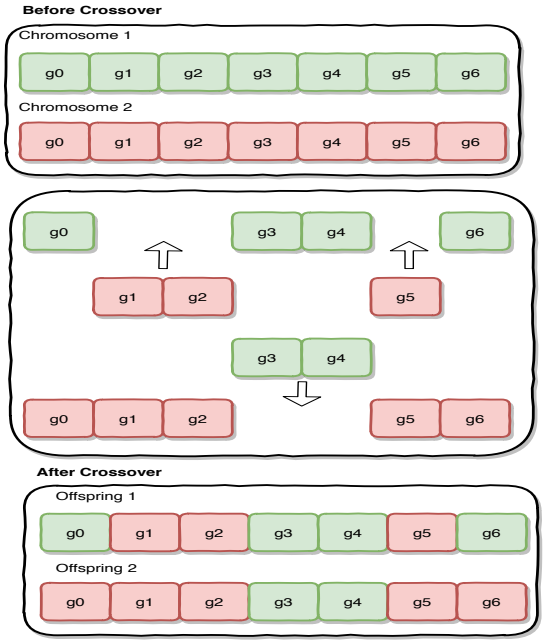


Fig. 2: An example of the elastic crossover operator

denote j^{c+n} as the random gene between any genes inbound the N length, and new descendant chromosomes are denoted as $\mathcal{C}_{(M+1)}$ and $\mathcal{C}_{(M+2)}$ respectively. The new offsprings are ultimately produced by changing random parent genes as depicted in (14b). Obviously, the quality of generated children may be worse than their ancestors or the duplication of solutions may happen at different parallel levels. The random genes of each parental chromosome j^{c+n} are randomly selected, and they can be even the first or last gene in the parents' chromosomes because the mated children are apparently based on

TABLE I: *Compared Algorithms*

Notation	Description
DPGA	Novel Distributed Parallel GA-based Algorithm proposed in this paper
TGAL	Typical GA-based Algorithm for Link Mapping [12]
NTANRC-S	Network Topology Attribute and Network Resource-considered algorithm (Stable) [10]
G-SP	Greedy Node Mapping with Shortest Path Based Link Mapping [14]
R-ViNE	Random Node Mapping with Shortest Path Based Link Mapping [14]
D-ViNE	Deterministic Node Mapping with Shortest Path Based Link Mapping [14]

distinct genes rather than a successive sequence of genes. Furthermore, we might wonder how many genes are chosen to exchange because if all genes are selected, two parental chromosomes are just swapping all genes each other. In this case, our crossover stage is veritably dysfunctional. As a result, we define an elastic factor e^f which determines maximum number of genes that can be changed at each chromosome. It can be considered as a proportion of the N length. For instance, if $e^f = 0.75$ (75%) and $N = 7$, the number of genes that can be randomly chosen for mating is ranging from $[1 - 5]$

Mutation: typically implements random modifications on children to produce new offspring. Mutation is implemented in the purpose of sampling the solution space and broadening the search. This is a crucial piece of the solution process that can somehow prevent from falling into the local optima. A mutation operator usually includes a mutation point denoted as j^m that is generated in random. At this point, a new gene which is selected from the original path pool can substitute any random existing gene in the in-processed chromosome to create a new child. Definitely, the newly selected gene must pass the feasibility check. We denote j^m and $g_{r'}^{j^m}$ as the mutation point and the new gene that replaced the existing one in $\mathcal{C}_{(M+1)}$ respectively. After substitution, the mutation solution $\mathcal{C'}_{(M+1)}$ can be described as $\mathcal{C'}_{(M+1)} = [g_s^1 \cdots g_{r'}^{j^m} \cdots g_s^N]$.

3) Solution Sorting and Terminations

A GA-based mapping process conducting at each slave node is expectantly terminated whenever reaching a predefined number of iterations. After the sorting step that is based on the fitness function values, the best mapping solution among the feasibles is selected to convey to the synchronization for the global ranking. In addition, a parallel operation comprises a series of concurrent processes, and each will finish its assigned work at different time. Waiting until the last process accomplishing its particular task is actually painful, but it does not always guarantee to achieve the envisaged outcome. In some unexpected situations where a specific process takes excessively longer time to accomplish, this situation influences on the total operation time. Moreover, two or more succeeding processes might be jammed because they need to wait until such process completes its work (e.g. deadlock). Hence, the master procedure in our parallel model will be finished if the best solution for VLiM found has not consecutively improved during t times, where t is known as a termination parameter.

4) Synchronization and VNR allocation

This step is premeditatedly aimed at determining the final VNE solution for the VLiM request by a ranking process based on fitness function values of the feasible solutions that have received from slave nodes. Consequently, the corresponding VNR would be accepted and allocated onto the physical network relied on the information of the related node and link mapping solutions. Accordingly, the residual network resources will be updated in advance.

IV. Performance Evaluation

A. Simulation setup

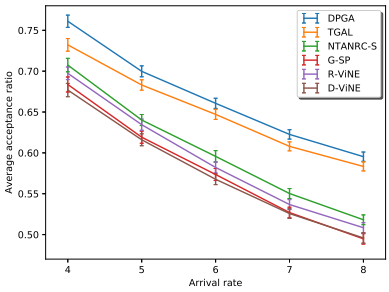
We have deployed a discrete-event simulator to evaluate the proposed GA algorithm with same parameter settings as [3]. Accordingly, SN and VNs are generated using a popular GT-ITM topology generator [15]. In our simulation, substrate networks are configured with 50 nodes, randomly placing on a 25×25 Cartesian plane. They are randomly connected to average 140 edges applying the Waxman model with $\alpha = 0.5$ and $\beta = 0.2$. Fundamentally, α determines the maximal edge probability whilst β basically defines the edge length. Likewise, CPU and bandwidth resources of the SN are uniformly generated in between 50 and 100 while the VNRs arrive in network following the Poisson process with an average rate ranging from 4 to 8 virtual networks per 100 time units. The lifetime of VNRs obeys an exponential distribution with an average value of $\mu = 1000$ time units. Besides, the number of virtual nodes for each VN graph is identified by a uniform distribution between 2 and 10 in random with an average graph connectivity at 50%. Furthermore, the CPU capacity of the virtual nodes and the bandwidth requirements of the virtual links are integers uniformly distributed between 0 to 20 and 0 to 50 respectively. Similar to [14], we set $w_b = w_n = 1$ in this paper. Simulations are running for $50,000$ time units which is exceptionally longer than the average lifetime of a VN 50 times. This simulation time is actually long enough to achieve a large number of independent samples. Moreover, all evaluation figures with average values are plotted with 95% confidence interval.

B. Comparison Methods

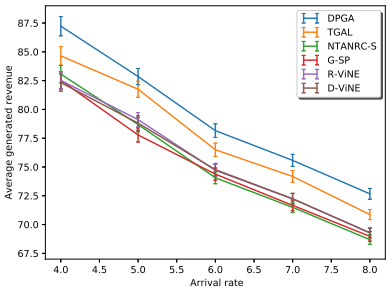
In our performance evaluation, we compare the proposed GA-based algorithm with the selected competitors as listed in Table I. These are intentionally taken into account because of the fact that [3] is certainly one of the most popular research paper in VNE field. We select DViNE, RViNE, and G-SP for comparison because of two objectives: performance and speed. In details, D-ViNE and R-ViNE algorithms frequently provision the great performance by implementing a relaxed linear programming approach for node embedding problems. On the other hand, G-SP applies the shortest path mechanism for link mapping, and it is extensively conducted by either heuristic or meta-heuristic approaches as discussed in section V. It is notably to emphasize that the shortest path algorithm is widely accepted as the fastest link embedding algorithm in VNE due to its simplicity. In order to prove the effectiveness of our proposed algorithm, the state-of-the-art VNE algorithm, namely NTANRC-S, based on node ranking method is selected for comparison. NTANRC-S algorithm is preferred because it shows better performance than NTANRC-D in [10]. Moreover, we select TGAL [12] since it is one of the first algorithms deliberately designed for link mapping stage, and it is primarily based on the typical GA-based algorithm running in parallel. By comparing with TGAL, we validate the improvement of our proposed GA-based algorithm in this paper.

C. Evaluation Results

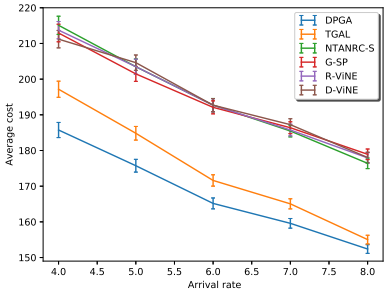
As shown in Fig. 3 and Fig. 4, DPGA algorithm achieves higher acceptance ratios by accepting more VNRs with significantly less average costs than all competitors as depicted in Fig. 3c, which obviously leads to much higher revenue for our proposed algorithm (Fig. 3b). Specifically, DPGA performs better than R-ViNE - the best performance in [3], NTANRC-S and TGAL for approximately 9.14% (17.1%), 7.58% (14.91%) and 3.95% (2.1%) at the acceptance ratios of 4 (8) respectively as represented in Fig. 3a. In this paper, we substantially concentrate on the virtual link mapping problem, so the average remaining bandwidth of all compared algorithms is selected and illustrated in Fig. 4b. When more VNRs arrive, the link utilization of compared algorithms unsurprisingly increase. However, our link mapping result achieves superior performance since it consumes less bandwidth to embed the link mapping requests, which is confirmed by Fig. 4a and 4b. More remaining bandwidth means higher probability of accepting new VNRs. DPGA and TGAL obtain much better average path length metric compared to other algorithms as shown in Fig. 4a, which contributes to lower costs and better remaining bandwidth results. This is because we consider the hop-count factor in FF to determine the VNE link embedding solution. In addition, the average CPU execution time of DPGA embedding a VNR is almost the same as TGAL, but our proposed algorithm attains an absolute faster speed than G-SP for more than 43% as illustrated in Fig. 4c which had been widely accepted as the fastest and most popular VLiM algorithm. It is observed that NTANRC-S nearly performs the same execution time in comparison with G-SP as it also deploys the node-ranking heuristic algorithm for VNoM and the shortest path method for VLiM. The remarkable performance is indeed gained because our proposed GA-based approach explores the searching space efficiently in order



(a)

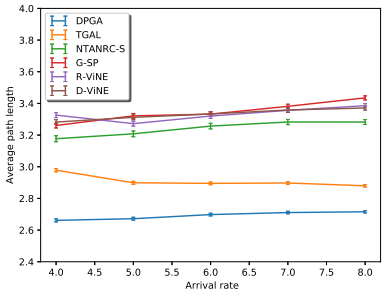


(b)

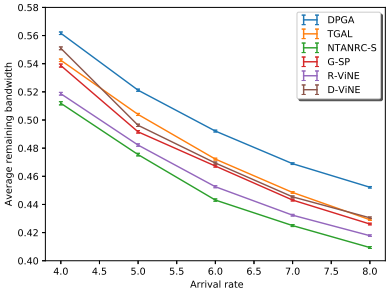


(c)

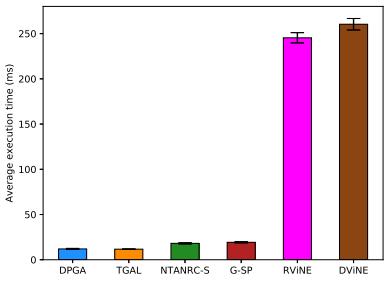
Fig. 3: **(a)** VNR Acceptance Ratio **(b)** Average generated revenue **(c)** Average cost of accepting VNRs



(a)



(b)



(c)

Fig. 4: **(a)** Average path length **(b)** Average remaining bandwidth **(c)** Average CPU execution time

to acquire more feasible link embedding solutions due to a novel crossover operator. In addition, our fitness function guides the GA-based algorithm on the-right-track by minimizing the total embedding cost, propagation delay and hop-count factors. Furthermore, our novel GA algorithm tends to evaluate various feasible solutions in a greatly less time consumption due to a proper parallel operation paradigm as depicted in Fig. 4c. Accordingly, the time complexity of our GA-based algorithm is rapidly reduced to logarithmic $O(log(p))$. Regarding the execution time study for time complexity and convergence, interested readers may refer to the paper [12] for further theoretical analysis.

V. Related Work

With demanding research endeavors on NV, [14] essentially provides a comprehensive survey to this research field. VNE problem is $\mathcal{NP}$-hard in nature, which is intractable to solve utilizing Integer Programming (LP). Most researched work in VNE is almost focused on looking for efficient heuristic algorithms due to the computational complexity of extract methods. A coordinated node and link approach for the node embedding relaxing the intractable integer constraints, and then taking advantage of rounding techniques to achieve the optimal node mapping was first introduced in [3]. Huang et al. in [5] primarily extended [3] to enable a novel node splitting scheme and node collocation. Node ranking approach [10] that was inspired from Google PageRank algorithm was proposed to rank virtual and substrate nodes for each VNR, which is based on three topology attributes and global network resources. In the meantime, one of the most popular Artificial Intelligence (AI) approaches, Genetic Algorithm, formally applied to the VNE problems was first reviewed in [6] and [7]. The research work [6] proposed node ranking methods relied on GA algorithms with various topology attributes. In addition, [7] ultimately conducted an evaluation comparison amongst Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and GA where the node mapping stage was intentionally considered. Additionally, a VNE model that is relied on GA algorithm to resolve the VNE problems in multiple InP domains was examined in [8] whereas a research work [9] reordered the mutation operator in the traditional GA algorithm to produce higher quality offspring in the initial population. Recently, a parallel GA-based algorithm in [12] was deliberately devised for the link mapping phase.

VI. Conclusion

NV is essentially a key factor of the foreseeable success of the prospective network architectures (e.g. 5G-and-beyond networks, virtualised IoT networks), so an efficient resource allocation algorithm for the VNE applications is highly desirable. In this paper, we proposed an intelligent parallel algorithm based on a novel GA algorithm for online VN link embedding, considering both scalability and optimality. Particularly, we redesign the crossover operator and present a multi-constrained fitness function considering multiple network attributes guiding GA algorithm towards an efficient VNE solution. Our proposed GA-based algorithm eventually outperformed state-of-the-art algorithms in all evaluation matrices including performance and time efficiency. 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] 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.

[5] 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.

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

[7] X. Chang, X. Mi, and J. Muppala, "Performance evaluation of artificial intelligence algorithms for virtual network embedding," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 10, pp. 2540 – 2550, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197613001358

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

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

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

[12] 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] C. Molnar and J. Gair, "Concepts of biology - 1st canadian edition," *BCcampus*, May 2015. [Online]. Available: https://opentextbc.ca/biology/

[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] 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.

Rethinking Virtual Link Mapping in Network Virtualization

Khoa TD 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*—Virtual Network Embedding (VNE) that addresses the embedding problems of heterogeneous virtual networks onto a physical limited-capacity infrastructure efficiently is a major challenge in ~~NV~~network virtualization (NV). VNE is computationally intractable when considering various constraints on nodes and links, and is also known as $\mathcal{NP}$-hard even in offline embedding. Although the VNE problems have received much attentions over recent decades with a vast number of VNE solutions, the majority of them only focus on VNE node mapping, whilst leaving the link mapping stage for the shortest path method or multicommodity flow (MCF) algorithm. We persuasively argue that node and link mappings equally play pivotal roles to approach an efficient VNE solution. In this paper, we reassess the role of link mapping stage in VNE problem, and then propose a novel intelligent VNE orchestration which effectively implements a distributed parallel model to reduce the operation time remarkably. Extensive evaluation results show that our proposed ~~algorithms are~~ algorithm is not only faster than ~~the most popular~~ state-of-the-art VNE algorithms in speed, but also better in all performance metrics.**

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

I. Introduction

~~Network virtualization~~ NV is considered as a promising paradigm to pave the success of the future generation networks such as 5G-and-beyond [1], virtualised IoT networks [2]. NV allows to share the physical network resources among multiple virtual network requests (VNRs), enabling an isolated coexistence of multiple virtual networks (VNs) on a single ~~physical~~ substrate network (SN). This key technology brings more efficient resource utilization to the ~~substrate network,~~ SN and offer a great opportunity of implementing as well as evaluating new network protocols or architecture designs~~, and eventually~~. It also prevents an unnecessary expansion of network infrastructure.

~~In VN environment, a service provider (SP) typically converts a requested service/application into a virtual network, 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. To gain expected profits, InPs desire an efficient resource allocation scheme that can substantially reduce the embedding costs and increase revenue by serving as many VNRs as possible. Generally, a VNR that includes a set of nodes connected via links to construct a specific topology dynamically arrives and resides in the network during a random duration in most real scenarios.~~ The critical challenge of embedding VNRs with diverse topology and stringent resources onto the underlying physical network is ~~also~~ known as Virtual Network Embedding (VNE) problem.

In essence, VNE process that enables mapping requested VNs onto the underlying shared physical network can be decoupled 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 ~~even in unsplittable scenario [3].~~ [3]. However, it is ~~importantly~~ important to note that VLiM problem is de facto more challenging than the analogue VNoM due to the requirements that all the substrate links along which a virtual link is mapped onto must have enough residual capacities to support the bandwidth requirement of the virtual link, which results in the bandwidth fragmentation problem more likely to occur. In practise, the most common failures of mapping VNRs invariably emanate from the ineffective link mapping algorithm [4]. Consequently, we presume that an appropriate design of VLiM mechanism will not only improve the efficiency of link resource utilization, but also increase the number of accepted VNRs.

In fact, the formulated optimization models such as Integer Linear Programming (ILP) are commonly proposed to achieve optimal VNE solutions, but they cannot be actually tailored for online VNE problems due to their intricacy, non-scalability and non-polynomial time issue. Instead of exact methods, most of research papers have simply adopted light-weighted heuristic algorithms to tackle the aforementioned impediments of the formulated optimization models. However, most research work [3]–[10] engrosses in seeking an efficient node mapping, but it seems to underestimate the link mapping stage since they completely entrust to ~~only~~ k-shortest path or multicommodity flow (MCF) algorithms, which definitely restricts VNE link mapping options.

Towards 5G-and-beyond networking and smart IoT, VNE problem where the physical infrastructure allows splittable and unsplittable resource configurations is indisputably an essential research topic in Software Defined Network (SDN), Network Function Virtulization (NFV) and Future Edge Clouds. Splittability permits a virtual link demand to be embedded on multiple substrate paths whilst if it is mapped onto a single physical path with fixed node mapping, which reduces to unsplittable configuration. Although splittable-based embedding is literally expected to ~~gain~~ obtain better resource utilization, it may generate a larger overhead to consistently maintain the network state [11]. Due to the aforementioned reasons, we merely focused on unsplittable mapping in this paper. In contrast, parallel algorithms can be tremendously beneficial in dealing with the intricate computing tasks thanks to lower hardware costs for computing recently.

In this paper, we ~~present a novel intelligent algorithm based on an enhanced Genetic Algorithm (GA)~~ propose a novel GA-based VNE algorithm that is relied on new design of the crossover operator for VLiM. The multi-constrained fitness function considering the embedding cost, hop-count and propagation delay has driven the proposal to an efficient VNE algorithm. Our proposed algorithm, that exploits a set of distributed parallel machines~~to address the VNE link mapping problems.~~

, enables to embed multiple link mapping requests at the same time so as to reduce the execution time. To the best of our knowledge, this is the first paper that applies an elastic crossover mechanism in GA algorithm to VNE problems. This paper is an extension of [15] towards the crossover operator and the improved fitness function, which not only achieves better performance than [15], but also outperforms state-of-the-art VNE algorithms.

The remainder of this paper is organized as follows: the network model is formulated in Section II and then we present our proposed distributed parallel GA-based algorithm for VNE link mapping in Section III. The performance evaluation is introduced in Section IV whilst the related work is presented in Section V. Section VI is finally a conclusion of this paper.

II. Network Model and Problem Descriptions

A. Virtual Network Assignment

The VNE substrate network is modelled as a weighted undirected graph $G^s = (N^s, L^s)$, in which N^s is the set of all substrate nodes and L^s is the set of all substrate links. Basically, each substrate node $n^s \in N^s$ that has a geographic location $loc(n^s)$ is characterised by the available CPU capacity $c(n^s)$, whereas each substrate link $l^s \in L^s$ between any two substrate nodes has a finite bandwidth capacity $b(l^s)$. Memory and storage resources will be omitted in this article for simplification. In VNE research, we can model the i^{th} arriving VNR as a weighted undirected graph denoted as $G_i^v = (N_i^v, L_i^v)$, where N_i^v is the set of all virtual nodes and L_i^v is the set of all virtual links towards the i^{th} VNR. Each virtual node $n_i^v \in N_i^v$ is inherently characterised by a requested CPU capacity $c(n_i^v)$, whilst 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 has a requested bandwidth capacity $b(l_i^v)$. ~~Additionally, each~~ Each VNR has a preferable mapping radius $D(n_i^v)$ that discloses how far a virtual node n_i^v can be placed from its location identifier $loc(n_i^v)$. Mapping ~~a VN request~~ the i^{th} VNR G_i^v onto the ~~substrate network~~ SN G^s can be decomposed into two main components as determined above: ~~Virtual Node Mapping (VNoM) and Virtual Link Mapping (VLiM)~~VNoM and VLiM. Under node mapping stage, a virtual node from a ~~VN request~~ VNR can be embedded onto a substrate node $\mathcal{A}_\mathcal{N} : N_i^v \to N^s$, with $n^v \in N_i^v$ subject to:

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

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

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

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

where $n^v \to n^s$ defines the virtual node n^v that is mapped on the substrate node n^s, and the distance between the geographical locations of node i^s and j^d is measured by $\mathcal{D}(i^s, j^d)$. Besides $R_N(n^s)$ denotes the residual/available CPU capacity of a substrate node. In fact, a virtual link is mostly embedded on the corresponding substrate path with one or more substrate links. As such, this unsplittable link embedding can be denoted by $\mathcal{A}_\mathcal{L} : L_i^v \to L^s$ whilst $l_i^v = (s_i^v, d_i^v) \in L_i^v$, $\mathcal{E}^s(\mathcal{A}_\mathcal{L}(l_i^v))$ is a set of all possible substrate paths from source node $\mathcal{A}_\mathcal{N}(s_i^v)$ 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:

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

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

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

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

From the InPs' perspective, the main VNE objective is to maximize their accumulated revenues while keeping its embedding cost minimal. In this paper, the generated revenue of InPs is practically calculated as the sum of total virtual resources embedded on the ~~substrate network~~ SN over time. Accordingly, the revenue of i^{th} VNR G_i^v is computed as below:

$$\mathcal{R}(G_i^v) = w_b * \sum_{l_i^v \in L_i^v} b(l_i^v) + w_n * \sum_{n_i^v \in N_i^v} c(n_i^v) \tag{9}$$

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_b and w_n are the unit weights of the mapped bandwidth and CPU resources respectively.

Cost: we likewise characterize the cost of the i^{th} VNE $C(G_i^v)$ as the sum of total network resources allocated to the i^{th} VN.

$$C(G_i^v) = \sum_{n_i^v \in N_i^v} c(n_i^v) + \sum_{l_i^v \in L_i^v} \sum_{l^s \in L^s} f_{l^s}^{l_i^v} \tag{10}$$

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

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

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:

$$\mathcal{R}_m(L^s) = \sum_{l^s \in L^s} (b(l^s) - \sum_{l_i^v \to l^s} b(l_i^v)) \tag{12}$$

Meanwhile, there are new VNRs arrived, the InP will intrinsically calculate the residual network resources, and then attempts to embed the corresponding VNRs onto the physical network depending on such achieved remaining resource information. ~~Subsequently, higher~~ Higher remaining bandwidth would bring higher chance of accepting the prospective virtual networks.

Fitness Function (FF): the fitness values of each solution ~~crucially~~ determine which one will reproduce and remain "alive" in the next generation, relevant to the predefined objectives to be optimized in our proposed GA-based algorithm in Section III-B. As a result, this function is utilized to examine the quality of each VLiM solution among several feasible ones so that its values can provide a scientific proof for electing the corresponding solutions in GA stages. ~~Specifically~~In details, we take the cost of embedding a VNR into consideration in this paper, so solutions with less cost generated are definitely preferable. Moreover, we consider hop-count as an important factor into FF as it is substantially associated with bandwidth consumption. This means that less hop-count solution would consume less bandwidth, and then leaves more residual network bandwidth, increasing the possibility of the upcoming VNRs being accepted. The propagation delay of VLiM solutions is also estimated and added into FF as another constraint accompanying with the hop-count attribute in order to construct a multi-constrained fitness function. Fitness function $\mathcal{F}(\mathcal{S}_z)$ is eventually calculated as below:

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

where, ~~$\mathcal{S}_z$~~$\mathcal{S}$, h and $d_\mathbb{P}$ are a ~~z^{th}~~ 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. Distributed Parallel Resource-Allocation Algorithm

A. Backgrounds and Ideas

Parallel and distributed computing has recently emerged as an effective mechanism to tackle large and complex problems with less time consuming and lower cost by supporting the concurrency. In addition, GA algorithm is an appealing AI approach for dealing with both constrained and unconstrained optimization problems by adopting the natural selection idea. ~~conducted as a parallel search without considering dependency among the feasible solutions because they are mutually exclusive. Typically, Genetic Algorithm~~ ~~Moreover, [12] proved that GA can be~~

A typical GA consists of four primary operators: initialization, selection, crossover and mutation. Crossover operator can be recognized as an exploitation phase in which the global optimum is positively expected to discover, so we argue that a creative crossover operator ~~will increase~~ would improved the efficiency of our GA algorithm ~~along~~

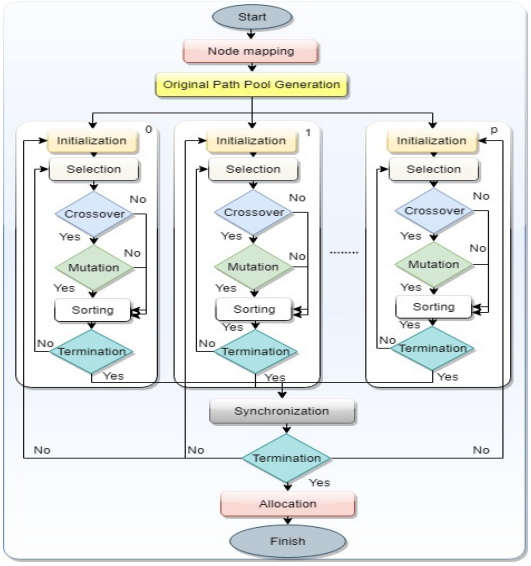


Fig. 1: *Parallel operation scheme*

~~with a proper fitness function~~driven by an appropriate FF. Inspired from the DNA replication process in [13], this mechanism allows cell division to occur. A ~~parent~~ parental cell can be split into two or more daughter cells by this process. They can ~~complicatedly~~ inherit diverse combinations of partial or all DNA. In this research work, we redesign a novel crossover operator for the proposed GA algorithm. This new mechanism allows to proportionally exchange the random genes between the parental chromosomes to generate new offsprings. ~~Furthermore, as argued in previous sections~~

As discussed, we assume that the role of link mapping stage in VNE is being underestimated, and the ~~unbeatable~~ fastest speed of shortest path method can be ~~broken due to a proper parallel implementation~~ vanquished by implementing an ingenious parallel operation scheme. Thus, we propose a novel intelligent GA-based orchestration algorithm for VLiM stage, ~~operating~~ running on a predefined number of independently distributed parallel machines ~~where they are independently running~~ (e.g. virtual machines) to generate the feasible solutions denoted as chromosomes. To prove our hypothesis, we deploy a simple Greedy node mapping as the same with G-SP algorithm in [14] due to its simplicity. This selection can not only guarantee the rapid embedding speed, but also maximize the residual network resources leading more successfully allocated node mapping requests in future. Our proposed parallel GA scheme is ~~present~~ presented in Fig 1.

B. Distributed Parallel Genetic Algorithm (DPGA)

As depicted in Fig 1, we ~~basically~~ present the functioning procedures sequentially working under a single master node such as node mapping, ~~synchronization, etc.~~original path pool generation, synchronization, allocation, whereas the ~~other ones~~ others handle the parallel GA algorithms to ~~achieve~~ find the feasible solutions for VNE link mapping working as several slave nodes. Each slave machine is independently running with defined iterations, and then the best-matching feasible VLiM outcome ~~will be selected among~~ is selected amongst the parallel machines. Unlike other research papers ~~that sequentially embed requested virtual links one-by-one~~embedding virtual link requests sequentially, our proposed algorithm enables to ~~embed~~ map all link requests of a ~~virtual network together~~VNR altogether. A chromosome $\mathcal{C}_f$ including several genes g_i^j denotes a feasible link embedding solution for all virtual link requests of a VNR~~whereas a~~. Each gene g_i^j ~~is associated with a physical path~~, where i and j indicate its current chromosome ~~as well as~~ and virtual link respectively~~.~~, is associated with a substrate path that is a feasible solution for a virtual link request. It means that the number of genes corresponding to a VNR constitute a chromosome.

1) Initial path pool generation

We deliberately create the potential path database for the requested ~~mapping~~ virtual links before conducting link mapping procedures~~operated~~. For each pair of source-destination, a k-shortest path algorithm e.g. Dijkstra's algorithm is simply deployed to identify k-shortest paths ~~for~~ during the path pool generation. ~~It is argued that this~~ This intrinsic process can be obviously determined prior to the arrival of online VNRs since the substrate network is static.

2) Slave node

Population Initialization: each slave machine usually begins to ~~conduct~~ handle the proposed GA algorithm with a population initialization step where each chromosome $\mathcal{C}_f$ defines a feasible solution. It is assumed that there are M chromosomes and each chromosome has N genes. An initial population $\mathcal{P}$ (MxN size) at the k^{th} machine can be represented as below:

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

To construct a chromosome, each gene associated with the potential embedding solution of a virtual link request shall be uniformly selected from the initial path pool in random. ~~However, this~~ This solution must pass the feasibility-checking process to ~~generally~~ become a potential link solution. This checking procedure is actually necessary since it guarantees the underlying SN still has adequate residual resources to embed the corresponding request. All N potential genes that have already passed the feasibility check will compose a chromosome, considering as a feasible solution for the VLiM.

Selection: ~~determines the chromosome individuals as parents for the upcoming crossover operation. Basically, more than one pair of parent chromosomes can be intentionally selected from this step. Furthermore,~~ in order to enhance the parallelism degree, the parents are randomly chosen from the initial population with replacement. We ~~also~~ apply the fitness-based proportionate selection scheme to adopt parents from the initial population, which conceptually relies on the cumulative sum of the fitness relative weights (13). However, the children produced might either achieve better or worse quality than their parents in next operators due to the randomness selection.

$$\mathcal{P} = \begin{bmatrix} \mathcal{C}_1 \\ \vdots \\ \mathcal{C}_s \\ \vdots \\ \mathcal{C}_r \\ \vdots \\ \mathcal{C}_M \\ \mathcal{C}_{M+1} \\ \mathcal{C}_{M+2} \end{bmatrix} = \begin{bmatrix} g_1^1 & \cdots & \cdots & g_1^{j^{c+2}} & g_1^{j^{c+3}} & \cdots & \cdots & \cdots & g_1^N \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ g_s^1 & \cdots & \cdots & \cdots & g_s^{j^{c+3}} & g_s^{j^{c+4}} & \cdots & \cdots & g_s^N \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ g_r^1 & \cdots & g_r^{j^{c+1}} & g_r^{j^{c+2}} & \cdots & \cdots & g_r^{j^{c+5}} & \cdots & g_r^N \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ g_M^1 & \cdots & \cdots & g_M^{j^{c+2}} & g_M^{j^{c+3}} & \cdots & \cdots & \cdots & g_M^N \\ g_s^1 & \cdots & g_r^{j^{c+1}} & g_r^{j^{c+2}} & \cdots & \cdots & g_r^{j^{c+5}} & \cdots & g_r^N \\ g_r^1 & \cdots & \cdots & \cdots & g_s^{j^{c+3}} & g_s^{j^{c+4}} & \cdots & \cdots & g_s^N \end{bmatrix} \tag{14b}$$

Crossover: this is literally considered as one of the most ~~significant~~ important step in GA algorithm, which primarily combines the parental chromosomes to produce new offsprings in next generations. We define $\mathcal{C}_s$ and $\mathcal{C}_r$ as the parental chromosomes with the corresponding indices s and r in the initial population. Different from typical GA crossover operator that normally employs one or more static random crossover points to exchange the consecutive sequence of genes between two parental chromosomes, we ~~separately~~ proceed the crossover phase on selected chromosomes with different sequences of random numbers. Specifically, each related child has a different number of random genes to exchange with the ~~other~~ others and vice versa. For example, Fig 2 illustrates our proposed crossover operator where the 1^{st} chromosome exchanges the 1^{st}, 2^{nd} and 5^{th} genes with the other. On the other hand, the 2^{nd} chromosome gets new 3^{rd} and

4^{th} genes from the first chromosome. This novel operation will provide an elastic crossover mechanism producing resilient offsprings, which is particularly desirable to improve the exploitation phase. Moreover,

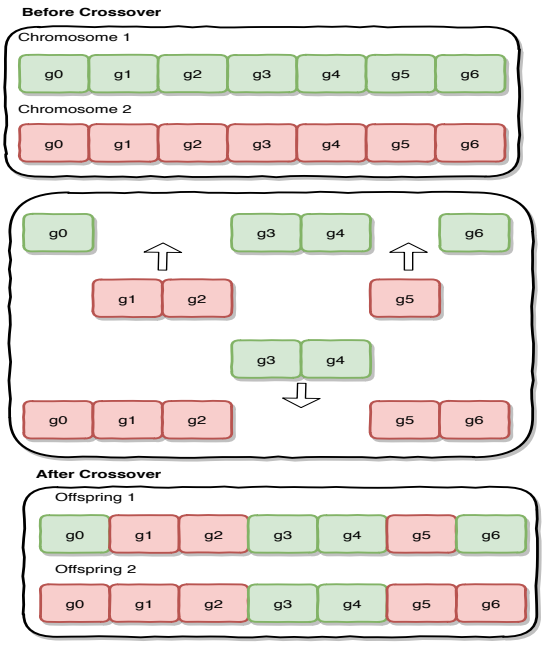


Fig. 2: An example of the elastic crossover operator

denote j^{c+n} as the random gene between any genes inbound the N length, and new descendant chromosomes are denoted as $\mathcal{C}_{(M+1)}$ and $\mathcal{C}_{(M+2)}$ respectively. The new offsprings are ultimately produced by changing random parent genes as depicted in (14b). Obviously, the quality of generated children may be worse than their ancestors or the duplication of solutions may happen at different parallel levels. The random genes of each parental chromosome j^{c+n} are randomly selected, and they can be even the first or last gene in the parents' chromosomes because the mated children are apparently based on distinct genes rather than a successive sequence of genes. Furthermore, we might wonder how many genes are chosen to exchange because if all genes are selected, two parental chromosomes are just swapping all genes each other. In this case, our crossover stage is veritably dysfunctional. As a result, we define an elastic factor e^f which determines maximum number of genes that can be changed at each chromosome. It can be considered as a proportion of the N length. For instance, if $e^f = 0.75$ (75%) and $N = 7$, the number of genes that can be randomly chosen for mating is ranging from $[1 - 5]$

Mutation: typically implements random modifications on children to produce new offspring. Mutation is implemented in the purpose of sampling the solution space and broadening the search. This is a crucial piece of the solution process that can somehow prevent from falling into the local optima. ~~Correspondingly, a~~ A mutation operator usually includes a mutation point denoted as j^m that is generated in random. At this point, a new gene which is selected from the original path pool can substitute any random existing gene in the in-processed chromosome to create a new child. Definitely, the newly selected gene must pass the feasibility check. We denote j^m and $g_{r'}^{j^m}$ as the mutation point and the new gene that replaced the existing one in $\mathcal{C}_{(M+1)}$ respectively. After substitution, the mutation solution $\mathcal{C}'_{(M+1)}$ can be described as $\mathcal{C}'_{(M+1)} = [g_s^1 \cdots g_{r'}^{j^m} \cdots g_s^N]$.

3) Solution Sorting and Terminations

A GA-based mapping process conducting at each slave node is expectantly terminated whenever reaching a predefined number of iterations. After the sorting step that is based on the fitness function values, the best mapping solution among the ~~feasible ones~~ feasibles is selected to ~~transfer to the next important component called~~ convey to the synchronization for the global ranking. In addition, a parallel operation comprises a series of concurrent processes, and each will finish its assigned work at different time. Waiting until the last process accomplishing its particular task is actually painful, but it does not always guarantee to achieve the envisaged outcome. In some unexpected situations where a specific process takes excessively longer time to accomplish, this situation influences on the total operation time. Moreover, two or more succeeding processes might be jammed because they need to wait until such process completes its work (e.g. deadlock). Hence, the master procedure in our parallel model will be finished if the best solution for VLiM found has not consecutively improved during t times, where t is known as a termination parameter.

4) Synchronization and VNR allocation

This step is premeditatedly aimed at determining the final VNE solution for the VLiM request by a ranking process based on fitness function values of the feasible solutions that have received from slave nodes. Consequently, the corresponding ~~VN request~~ VNR would be accepted and allocated onto the physical network relied on the information of the related node and link mapping solutions. Accordingly, the residual network resources will be updated in advance.

TABLE I: *Compared Algorithms*

Notation	Description
DPGA	Novel Distributed Parallel GA-based Algorithm proposed in this paper
TGAL	Typical GA-based Algorithm for Link Mapping [15]
NTANRC-S	Network Topology Attribute and Network Resource-considered algorithm (Stable) [10]
G-SP	Greedy Node Mapping with Shortest Path Based Link Mapping [14]
R-ViNE	Random Node Mapping with Shortest Path Based Link Mapping [14]
D-ViNE	Deterministic Node Mapping with Shortest Path Based Link Mapping [14]

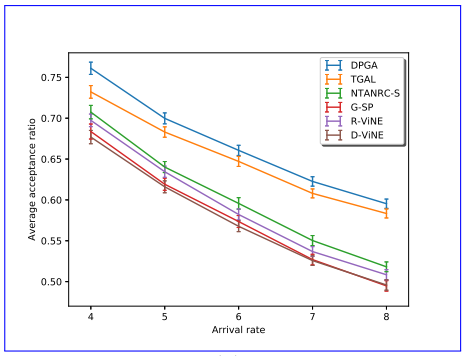
IV. Performance Evaluation

A. Simulation setup

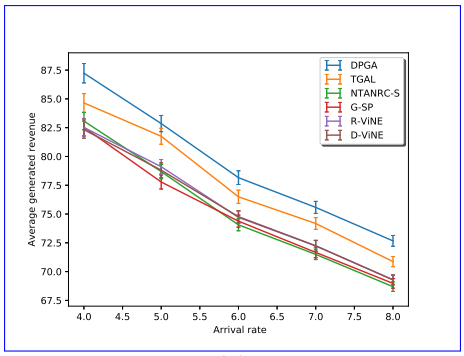
We have deployed a discrete-event simulator to evaluate the proposed GA algorithm with same parameter settings as [3]. Accordingly, SN and VNs are generated using a popular GT-ITM topology generator [16]. In our simulation, substrate networks are configured with 50 nodes, randomly placing on a 25×25 Cartesian plane. They are randomly connected to average 140 edges applying the Waxman model with $\alpha = 0.5$ and $\beta = 0.2$. Fundamentally, α determines the maximal edge probability whilst β basically defines the edge length. Likewise, CPU and bandwidth resources of the ~~substrate network~~ SN are uniformly generated in between 50 and 100 while the ~~VN requests~~ VNRs arrive in network following the Poisson process with an average rate ranging from 4 to 8 virtual networks per 100 time units. The lifetime of VNRs obeys an exponential distribution with an average value of $\mu = 1000$ time units. Besides, the number of virtual nodes for each VN graph is identified by a uniform distribution between 2 and 10 in random with an average graph connectivity at 50%. Furthermore, the CPU capacity of the virtual nodes and the bandwidth requirements of the virtual links are integers uniformly distributed between 0 to 20 and 0 to 50 respectively. Similar to [14], we set $w_b = w_n = 1$ in this paper. Simulations are running for $50,000$ time units which is exceptionally longer than the average lifetime of a VN 50 times. This simulation time is actually long enough to achieve a large number of independent samples. Moreover, all evaluation figures with average values are plotted with 95% confidence interval.

B. Comparison Methods

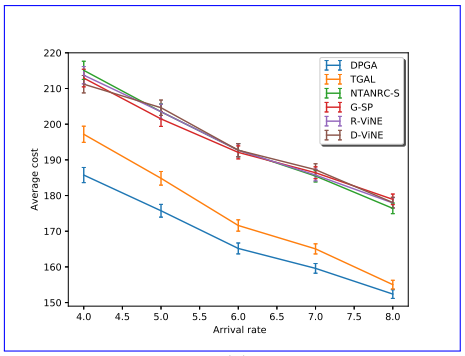
In our performance evaluation, we compare the proposed GA-based algorithm with the selected competitors as listed in Table I. These are intentionally taken into account because of the fact that [3] is certainly one of the most popular research paper in VNE field. We select DViNE, RViNE, and G-SP for comparison because of two objectives: performance and speed. In details, D-ViNE and R-ViNE algorithms frequently provision the ~~best~~ great performance by implementing a relaxed linear programming approach for node



(a)

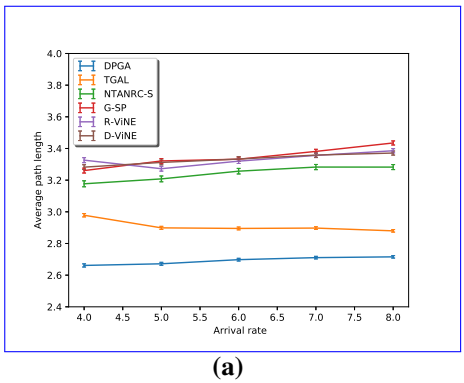


(b)

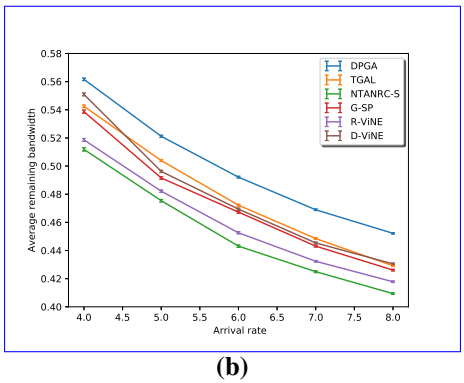


(c)

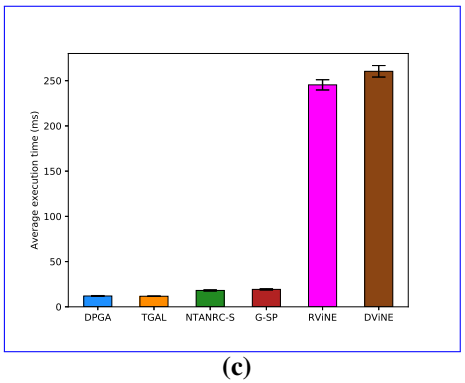
Fig. 3: **(a)** VNR Acceptance Ratio **(b)** Average generated revenue **(c)** Average cost of accepting VNRs



(a)



(b)



(c)

Fig. 4: **(a)** Average path length **(b)** Average remaining bandwidth **(c)** Average CPU execution time

~~mapping even in recent research~~embedding problems. On the other hand, G-SP applies the shortest path ~~method~~ mechanism for link mapping, and it is extensively conducted by either heuristic or meta-heuristic ~~methods~~ approaches as discussed in section V. It is notably to emphasize that the shortest path algorithm is widely accepted as the fastest link embedding algorithm in VNE due to its simplicity. In order to prove the effectiveness of our proposed algorithm, ~~we also~~ the state-of-the-art VNE algorithm, namely NTANRC-S, based on node ranking method is selected for comparison. NTANRC-S algorithm is preferred because it shows better performance than NTANRC-D in [10]. Moreover, we select TGAL [15] ~~to compare~~ since it is one of the first algorithms deliberately designed for link mapping stage, and it is primarily based on the typical GA-based algorithm running in parallel. By comparing with TGAL, we validate the improvement of our proposed GA-based algorithm in this paper.

C. Evaluation Results

As shown in Fig. 3 and Fig. 4, DPGA algorithm achieves ~~greater~~ higher acceptance ratios by accepting more VNRs with significantly less ~~total~~ average costs than all competitors as depicted in Fig. 3c, which obviously leads to much higher revenue for our proposed algorithm (Fig. 3b). Specifically, DPGA ~~is~~ performs better than R-ViNE - the best performance in [3]~~and TGAL in [15] for approximately 9% and 3% at different acceptance ratios~~ , NTANRC-S and TGAL for approximately 9.14% (17.1%), 7.58% (14.91%) and 3.95% (2.1%) at the acceptance ratios of 4 (8) respectively as represented in Fig. 3a. In this paper, we substantially concentrate on the virtual link mapping problem, so the average remaining bandwidth of all compared algorithms is selected and illustrated in Fig. 4b. When more VNRs arrive, the link utilization of compared algorithms unsurprisingly increase. However, our link mapping result achieves superior performance since it consumes less bandwidth to embed the link mapping requests, which is confirmed by Fig. 4a and 4b. More remaining bandwidth means higher probability of accepting new VNRs. ~~Moreover, similar to the parallel scheme used in [15] with a simple greedy method deployed for the node mapping stage due to its simplification,~~ DPGA and TGAL obtain much better average path length metric compared to other algorithms as shown in Fig. 4~~c~~ a, which contributes to lower costs and better remaining bandwidth results. This is because we consider the hop-count factor in FF to determine the VNE link embedding solution. In addition, the average CPU execution time of DPGA embedding ~~one~~ a VNR is almost the same as TGAL, but our proposed algorithm ~~achieves~~ attains an absolute faster speed than G-SP for more than 43% as illustrated in Fig. 4c which had been widely accepted as the fastest and most popular VLiM algorithm. It is observed that NTANRC-S nearly performs the same execution time in comparison with G-SP as it also deploys the node-ranking heuristic algorithm for VNoM and the shortest path method for VLiM. The remarkable performance is indeed gained because our proposed GA-based approach explores the searching space efficiently in order to ~~obtain~~ acquire more feasible link embedding solutions due to a novel crossover operator. In addition, our fitness function guides the GA-based algorithm on the-right-track by minimizing the total embedding cost, propagation delay and hop-count factors. Furthermore, our novel GA algorithm tends to evaluate various feasible solutions in a greatly less time consumption due to a proper parallel operation paradigm as depicted in Fig. 4c. ~~Regarding~~ Accordingly, the time complexity of our GA-based algorithm is rapidly reduced to logarithmic $O(log(p))$. Regarding the execution time study ~~on~~ for time complexity and convergence, interested readers may refer to the paper [15] for further theoretical analysis.

V. Related Work

With demanding research endeavors on NV, [14] essentially provides a comprehensive survey to this research field. VNE problem is $\mathcal{NP}$-hard in nature, which is intractable to solve utilizing Integer Programming (LP). Most researched work in VNE is almost focused on looking for efficient heuristic algorithms due to the computational complexity of extract methods. A coordinated node and link approach for the node embedding relaxing the intractable integer constraints, and then taking advantage of rounding techniques to achieve the optimal node mapping was first introduced in [3]. Huang et al. in [5] primarily extended [3] to enable a novel node splitting scheme

and node collocation. Node ranking approach [10] that was inspired from Google PageRank algorithm was proposed to rank virtual and substrate nodes for each VNR, which is based on three topology attributes and global network resources. In the meantime, one of the most popular Artificial Intelligence (AI) approaches, Genetic Algorithm, formally applied to the VNE problems was first reviewed in [6] and [7]. The research work [6] proposed node ranking methods relied on GA algorithms with various topology attributes. In addition, [7] ultimately conducted an evaluation comparison amongst Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and GA where the node mapping stage was intentionally considered. Additionally, a VNE model that is relied on GA algorithm to resolve the VNE problems in multiple InP domains was examined in [8] whereas a research work [9] reordered the mutation operator in the traditional GA algorithm to produce higher quality offspring in the initial population. Recently, a parallel GA-based algorithm in [15] was deliberately devised for the link mapping phase.

VI. Conclusion

NV is essentially a key factor of the foreseeable success of the prospective network architectures (e.g. 5G-and-beyond networks, virtualised IoT networks), so an efficient resource allocation algorithm for the VNE applications is highly desirable. In this paper, we proposed an intelligent parallel algorithm based on a novel GA algorithm for online VN link embedding, considering both scalability and optimality. Particularly, we redesign the crossover operator and present a multi-constrained fitness function considering multiple network attributes guiding GA algorithm towards an efficient VNE solution. Our proposed GA-based algorithm eventually outperformed state-of-the-art algorithms in all evaluation matrices ~~,~~ including performance and time efficiency. 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] 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.

[5] 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.

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

[7] X. Chang, X. Mi, and J. Muppala, "Performance evaluation of artificial intelligence algorithms for virtual network embedding," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 10, pp. 2540 – 2550, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197613001358

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

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

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

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

[13] C. Molnar and J. Gair, "Concepts of biology - 1st canadian edition," *BCcampus*, May 2015. [Online]. Available: https://opentextbc.ca/biology/

[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] 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.

[16] 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.

VTC2020 Rebuttal:

We are truly grateful for the insight reviews provided by the external reviewers of this manuscript. The precious comments and judgment on this study are clinically important to us to make this paper more valuable. In nature, nothing is perfect, and we are too. Despite our best efforts and intentions, we cannot always make it perfect.

Review 1:

Relevance

The manuscript proposes a Genetic Algorithm (GA)-based solution, named DPGA, for handling the Virtual Link Mapping (VLiM) problem in Virtual Network Embedding (VNE).

Even though the evaluation results show some sort of superiority of DPGA with respect to other state-of-the-art schemes, the novelty and contribution are not clearly identified, and the presentation needs work (see dedicated sections for these aspects).

Novelty

The novelty mainly lies in proposing a specific solution for VLiM, considering that most of literature in the context of VNE has been focused on the Virtual Node Mapping (VNoM) problem.

However, DPGA appears only slightly incremental with respect to the solution named TGAL, proposed by the same authors in a previous publication [15].

Contribution

Similar to "Novelty", the paper seems only slightly incremental with respect to [15]. TGAL in [15] is in fact already a GA-based solution for VLiM, and the only evident modification in the present paper is a slightly different crossover step for DPGA.

Is this the case? If not, the authors need to significantly rework their contribution, in order to better highlight what they propose as new in this paper.

Presentation

The presentation lacks of enough quality and clarity, and requires significant rework.

Many sentences are lengthy and difficult to follow, with many many adverbs and difficult constructs.

Many acronyms are not defined, please define all acronyms at their first appearance. Some acronyms are defined more that one time (not needed).

The difference between "splittable" and "unsplittable" is not clearly defined.

The mathematical notation appears uselessly complicated, with many subscripts, superscripts, and (re-)definitions probably not needed.

The description of the Fitness Function is unclear and difficult to parse.

Figures 1 and 2 are of poor quality.

A reference to a non-existing "(14b)" is given.

The GA description appears lengthy and can be probably improved. GA is a quite known optimisation approach, the paper should reference GA literature for detail, and focus on providing a clearer mapping between the GA definitions (chromosomes, genes, and functions) and the VLiM problem. This mapping appears often confused and difficult to understand.

From an architecture perspective, it is not clear how/where the distributed GA optimization is performed. Is it done at different physical or virtual machines? how the results across such machines are exchanged? what is the overhead, particularly in terms of communication? A

discussion on the complexity of the proposed approach with respect to the state-of-the-art could help to understand the overall scheme.
In Table I, a reference near to the benchmark algorithms would help.
It is unclear what is used for solving VNoM in the final proposed algorithm. As I see, DPGA is used to solve VLiM but a) the benchmark algorithms are complete algorithms solving both VNoM and VLiM (even though they solve VLiM with a simpler approach), and b) the performance analysis is given on the entire VNE process that is VLiM+VNoM, so: what DPGA uses to solve VnoM?
Figure 4a is not discussed in detail and is confusing, D-vine, R-vine and G-SP adopt a Shortest Path Based Link Mapping, so by definition they should provide the "shortest" paths, which does not seem the case.

Recommendation

In my opinion the paper does not meet IEEE VTC standards in its current shape.

Author' s response:

1 – Relevance:

The manuscript proposes a Genetic Algorithm (GA)-based solution, named DPGA, for handling the Virtual Link Mapping (VLiM) problem in Virtual Network Embedding (VNE).
Even though the evaluation results show some sort of superiority of DPGA with respect to other state-of-the-art schemes, the novelty and contribution are not clearly identified, and the presentation needs work (see dedicated sections for these aspects).

Thank you for pointing this out. We agree with this comment. We have highlighted the novelty and contribution of this paper in page 1, Introduction section, line [19-30] "*In this paper, ... VNE algorithms*".

2- Contribution

Similar to "Novelty", the paper seems only slightly incremental with respect to [15]. TGAL in [15] is in fact already a GA-based solution for VLiM, and the only evident modification in the present paper is a slightly different crossover step for DPGA.
Is this the case? If not, the authors need to significantly rework their contribution, in order to better highlight what they propose as new in this paper.

Thank you for your comments. In this paper, we introduce a novel elastic crossover operator for Genetic Algorithm (GA) and a multi-constrained fitness function which is based upon the embedding cost, hop-count and propagation delay factors to guide the GA algorithm towards an efficient VNE solution. To the best of our knowledge, this is the first time that an elastic crossover mechanism has been applied to VNE problems. To prove the effectiveness of our proposal, we also compare our proposed GA-based algorithm with state-of-the-art VNE algorithm, namely NTANRC-S in [10]. We also reveal these differences in page 1, line [19-30] "*In this paper, ... VNE algorithms*".

3 - **Presentation**

a - The presentation lacks of enough quality and clarity, and requires significant rework.
Many sentences are lengthy and difficult to follow, with many many adverbs and difficult constructs.

Many acronyms are not defined, please define all acronyms at their first appearance. Some acronyms are defined more that one time (not needed).

As shown in the revised version, we have fixed the editorial issues including long sentences, adverbs, complicated structures, acronyms, etc,.

b- The difference between "splittable" and "unsplittable" is not clearly defined.

We have clarified "splittable" and "unsplittable" terminologies in Page 1, line [9-12], “Splittability permits... unsplittable configuration”.

c- The mathematical notation appears uselessly complicated, with many subscripts, superscripts, and (re-)definitions probably not needed.

We have fixed “re-definitions” problem. For example: the cost function in equation (13). However, we believe that subscripts and superscripts should be remained in order to formulate our model. They are popularly used to formulate the VNE problems in many papers like [3-10].

d- The description of the Fitness Function is unclear and difficult to parse.

We have adjusted the fitness function, especially the equation (13) to make it clearer.

e- Figures 1 and 2 are of poor quality. A reference to a non-existing "(14b)" is given.

Thank you for your comments, the problems of figure 1, 2 and reference (14b) have been fixed.

f- The GA description appears lengthy and can be probably improved. GA is a quite known optimisation approach, the paper should reference GA literature for detail, and focus on providing a clearer mapping between the GA definitions (chromosomes, genes, and functions) and the VLiM problem. This mapping appears often confused and difficult to understand.

We have modified throughout several sections to make the descriptions of our proposed algorithm clear and precise such as section IIIA and section IIIB.

g- From an architecture perspective, it is not clear how/where the distributed GA optimization is performed. Is it done at different physical or virtual machines? how the results across such machines are exchanged? what is the overhead, particularly in terms of communication? A discussion on the complexity of the proposed approach with respect to the state-of-the-art could help to understand the overall scheme.
In Table I, a reference near to the benchmark algorithms would help.

We describe our proposed parallel architecture where GA algorithm performs at the link mapping stage to find feasible solutions for all virtual link mapping requests through figure 1 and section IIIB. To keep its simplicity, we overlook communications between slave-machines and all VNE processes have done in a single physical machine. Due to the limited length of paper we recommend the interested readers would reference paper [12] for more details about time execution analysis, convergence and time complexity.

h - It is unclear what is used for solving VNoM in the final proposed algorithm. As I see, DPGA is used to solve VLiM but a) the benchmark algorithms are complete algorithms solving both VNoM and VLiM (even though they solve VLiM with a simpler approach), and b) the performance analysis is given on the entire VNE process that is VLiM+VNoM, so: what DPGA uses to solve VNoM?

Thank you for pointing out. We utilize a simple node mapping mechanism that is based on Greedy algorithm due to its simplicity as cited in page 3, section IIIA, line [13-18], "To prove our hypothesis...in future". As discussed throughout this paper, DPGA merely focuses on virtual link mapping stage. We have eventually proved that just a simple node mapping (Greedy) combined with a complicated link mapping based on a proper architecture design will provide an efficient VNE solution that achieves both performance and speed perspectives.

i - Figure 4a is not discussed in detail and is confusing, D-vine, R-vine and G-SP adopt a Shortest Path Based Link Mapping, so by definition they should provide the "shortest" paths, which does not seem the case.

Thank you for your review. We have clarified Figure 4a in page 4, section IVC, line [58-62], "DPGA and TGAL...embedding solution". [3] is the most popular paper related to VNE problem with D-ViNE, R-ViNE and G-SP algorithms, which has been used as a benchmark in many research papers in this field. The shortest path problem is usually to find a path between two vertices in a graph such that a single network attribute can be merely optimized (e.g. delay, hop-count), and the shortest path algorithm is still considered as a heuristic algorithm. Although the shortest path mechanism performs quite well in both performance and speed, its solution can be optimal for a single virtual link request within a VNR, but it cannot obviously achieve a global optimum for several virtual link requests with multiple network constraints. As a result, this cannot guarantee an efficient VNE solution at the end, even we may deploy excellent node mapping algorithms (e.g. NTANRC-S in [10]). For that reason, we entitle our paper is "Rethinking Virtual Link Mapping in Network Virtualization" since we argue that node and link mappings equally play pivotal roles to approach an efficient VNE solution. Moreover, our proposed link mapping algorithm not only takes multiple objectives into account reflecting in the fitness function, but also processes several link mapping requests at the same time. We have eventually proved that just a simple node mapping (Greedy) combined with a complicated link mapping based on a proper architecture design will provide an efficient VNE solution that achieves both performance and speed perspectives.

Review 2:

We appreciate the positive feedback from the reviewer.

Relevance
Virtual Network Embedding (VNE) is an important issue in lots of scenarios. This article focused on unsplittable mapping in VNE and is relevant to this conference.

Novelty
Genetic Algorithm(GA) has a good effect in solving optimization problems. The authors applied the GA to VNE problem and got a good result. Whatsmore, this algorithm can embed all link requests of a virtual network together rather than sequentially embed requested virtual links one-by-one.

Contribution
The algorithm not only reduces the running time, but also reduces the cost and improves the profit.

Presentation
The article is clear in English and complete in structure.

Recommendation
This article is standard in writing and novel in the algorithm. It can be employed in this conference.

Review 3:

We appreciate the positive feedback from the reviewer.

Relevance
This paper studied virtual network embedding (VNE) problem.

Novelty
The proposed algorithm is based on a genetic algorithm using distributed parallel machines.

Contribution
This paper proposed an algorithm that solves the VNE mapping problem in a low delay.

Presentation
Clear presentation

Recommendation
Please refer the comments.

Comments

a- In problem (9), please explain how to determine the value of w_b and w_n. Also, what are the constraints of problem (9)?

   Thank you for pointing out. w_b and w_n are the weights of the mapped bandwidth and CPU resources respectively. From the InPs' perspective, they can define which factor will be preferable to achieve the desired revenue values. Similar to [3], we set w_b = w_n = 1 in page 4, section IVA, line 13.

b- It would be necessary to explain how problem (9) is different to the prior works. The contributions of this paper need to be emphasized.

   Thank you for your review. We agree with this comment. We have emphasized the novelty and contribution of this research work in page 1, Introduction section, line [19-30] "*In this paper, ... VNE algorithms*".

c- What is time complexity of the proposed algorithm? Please clarify if the algorithm converges.

   We identify the time complexity of our proposed algorithm in page 5, section IVC, line [7-11], "Accordingly...analysis". Due to the limited length of the paper we recommend the interested readers would reference paper [12] for more details about time execution analysis, convergence and time complexity.

d- In Fig. 4(a), the average path length of DPGA is the lowest. However, it is observed that, as the arrival rate increases, the average path length of DPGA increases, and the average path length of TGAL decreases. If the arrival rate is greater than 8.0, is it possible that the average path length of DPGA can be greater than the path length of TGAL?

Thank you for your question, it is actually interesting. This can be happened but we believe that DPGA is still better than TGAL since it advocates the population diversity. We leave this question for the future work.

Short list:
TrackChair ID number:
284-70830
Original track:
Spectrum Management, Radio Access Technology, Heterogeneous Networks