SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •
Special Focus on Mobile Network Virtualization

April 2017, Vol. 60 040308:1–040308:12
doi: 10.1007/s11432-016-9015-3

A model for virtual network embedding across multiple infrastructure providers using genetic algorithm

Isha PATHAK & Deo Prakash VIDYARTHI*

School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi 110067*, India*

Received August 1, 2016; accepted January 18, 2017; published online March 20, 2017

Abstract Network virtualization is an important aspect in cloud computing, where network is assumed to be consisting of infinite amount of nodes and links. The substrate network (physical network), has limited capacity and so virtual network embedding on the substrate network becomes a problem. Virtual network embedding is a computationally hard problem, considering various constraints on nodes and links. The proposed work applies Genetic Algorithm for the virtual network embedding problem for mapping multiple virtual network requests on infrastructure providers managing multiple substrate networks. Performance evaluation, through simulations, indicates that the proposed model preforms better for the performance metrics such as infrastructure provider revenue, acceptance ratio and node and link utilization in comparison to few other contemporary mapping models.

Keywords network virtualization, virtual network embedding, substrate network, NP-hard, genetic algorithm

Citation Pathak I, Vidyarthi D P. A model for virtual network embedding across multiple infrastructure providers using genetic algorithm. Sci China Inf Sci, 2017, 60(4): 040308, doi: 10.1007/s11432-016-9015-3

1 Introduction

To support multiple requests concurrently, virtualization of the resources is done. Cloud computing heavily relies on the virtualization because of two reasons; one - the system is transparent to the users, and two - number of resources is often less than the number of users in the system. Virtualization is done for various hardware resources such as storage, CPU, memory, etc. as well as software resources such as files, databases, etc. Network virtualization is a key to cloud computing and future internet design. It deals with the virtualization of the nodes and the links of the networks [1] resulting in logical network environments called virtual networks (VN). Virtual Network Embedding (VNE) problem is to map a virtual network, along with the resources they demand, onto the physical network. It deals with the embedding of virtual network request of a number of nodes and links onto the physical nodes and paths of the network. As such, the VN embedding problem is an NP-hard problem given various constraints on the nodes and links, even when virtual network requests arrive in an offline mode [2].

* Corresponding author (email: dpv@mail.jnu.ac.in)

An online virtual network mapping problem, in which VN requests are generated dynamically, is addressed in [3] using a sub-graph isomorphism backtracking method. In this approach, simultaneous node and link mapping is performed considering the constraints. On an inappropriate mapping decision, a backtrack to a recent valid mapping is applied which helps in avoiding re-mapping cost. In another work [4], a shared substrate network (SN) maps the virtual network topologies where the access-nodes are connected to one backbone node out of many that are star-connected. This algorithm is topology constrained as it works only for certain specific topologies. To reduce the complexity of the virtual network mapping, a cluster based distributed algorithm [5] where the virtual network requests can be decomposed in hub and spoke clusters (each cluster being mapped independently) is proposed. This method exhibits an inferior performance compared to any centralized algorithm. Zhu et al. [2] proposed a centralized algorithm to balance the load of the physical network. It maintains a balanced load on the nodes and links of the substrate network. Another mapping algorithm proposed by Yu et al. [6] considers that the resources on the substrate network are finite and allows path splitting along with path migration during the embedding. This might lead to fragmentation and may become a hurdle when one has to manage large virtual networks. Chowdhury et al. [7] proposed two VN embedding algorithms; Deterministic-ViNE and Randomized-ViNE. They solved the virtual network embedding problem in two stages; in the first stage virtual nodes are assigned to the substrate nodes and in the second stage virtual links are assigned to the substrate paths. The main drawback of this approach is the constraints on the nodes locations. Each end user is expected to specify the virtual node location which is unrealistic. It is because a specific region is associated with each virtual node where it could be hosted and if that region is not defined, it would be impossible to execute D-ViNE and R-ViNE. It performs better in comparison to few other heuristics such as [2, 6]. A totally different formulation based on Integer Linear Programming (ILP) is given in [8]. This ILP based mathematical formulation solves the virtual network embedding problem in a single step. A topology aware virtual network mapping solution is proposed by Butt et al. in [9] that consider reallocation and reassignment of virtual node and link respectively to avoid bottlenecks on the substrate network. The heterogeneity of virtual networks and a physical network is considered in a heuristic given by Nogueira et al. [10]. This heuristic is simulated on a small scale test-bed and achieves faster mapping with a mapping time in the order of tens of milliseconds. The mapping obtained is linearly scalable with the increase in the number of virtual networks.

In recent, evolutionary algorithms such as Ant Colony Optimization(ACO) [11], Particle Swarm Optimization(PSO) [12], Cat Swarm Optimization(CSO) [13], Genetic Algorithms(GA) [14] etc., have been effectively applied to many NP-class of problems such as scheduling problem, minimum weight triangulation problem, quadratic bottleneck assignment problem etc. Virtual Network Embedding (VNE) problem is a computationally complex problem and such evolutionary algorithms can be feasibly applied to this. Some of the works that apply GA, PSO and ACO algorithms, to solve the VNE problem, are as follows. A PSO based VNE algorithm, proposed in [15], achieves higher average revenue and VN acceptance ratio compared to the D-ViNE solution [7]. The QoS parameters in terms of bandwidth, power required and memory are optimized by introducing an embedding strategy based on ACO meta-heuristic for the virtual network embedding problem in [16]. GA has been extensively applied to solve complex constraint based optimization problems. VNE problem has been addressed using two GA based algorithms named CB-GA (based on cost and bandwidth) and RW-GA(based on markov random walk) derived on node ranking method [17]. The CB-GA and RW-GA result in better performance compared to PSO-based VNE algorithms in terms of average Infrastructure Provider (InP) revenue, acceptance ratio and revenue to cost ratio.

Most of the heuristics and meta-heuristics (some of which are discussed above) address the VN embedding problem for a single InP scenario. However, in a realistic scenario, multiple InPs managing heterogeneous domains are responsible to provision multiple VN requests to deliver end to end services. Multiple InP VNE problem addresses the issue in which a VN provider has to trade with more than one infrastructure providers for the resources in order to satisfy the VN requests. To the best of the authors' knowledge, this problem has not been addressed using GA, whereas GA has the potential to solve it effectively. The work, in this paper, focuses on applying a GA based meta-heuristic for this

problem to find a near optimal solution with an efficient allocation of the resources of the physical networks to the VNs and thus increase the revenue of the multiple InPs. The residual substrate network resources, serving the virtual networks, are also maximized with the proposed GA based VNE algorithm in a multiple InP network virtualization environment.

The contribution of the paper would be discussed further in detail, with Section 2 elaborating on the virtual network embedding problem. The proposed GA based virtual network embedding model (VNE-GA) is given in Section 3. Section 4 evaluates the performance through simulation and presents the results of the proposed VNE-GA algorithm. A comparative study, of the proposed model with other well-known VNE algorithms, is also given in this section. Finally, Section 5 concludes the work highlighting some future research directions for the VNE problem.

2 The virtual network embedding (VNE) problem

As mentioned earlier, the challenge of mapping all virtual networks (VN) along with their demands for the resources, on the physical network owned and managed by the InPs, frames the VNE problem. When the VN requests from the users are received, the VN providers cooperate with the multiple InPs and provision the VN requests by allocating available SN resources. This section describes in detail the virtual network embedding problem in a multiple InP scenario.

2.1 Virtual network request

A virtual network request is a data center application request by the user which demands some Quality of Service (QoS) and is represented as an undirected graph $G_V = (V, E, V_r, E_r)$, where V denotes the set of VN vertices, E denotes the set of VN edges, V_r denotes the amount of resources requested by the VN vertices and thus a particular amount of these resources from the InP are allocated to each virtual node $\mathrm{nv} \in V$. E_r denotes the resources requested by the VN edges and thus a definite amount of bandwidth is allocated to each virtual edge $\mathrm{ev} \in E$ for the exchange of data and information.

2.2 Substrate network

The physical network(s)/substrate network(s) serving the VN requests, can also be represented as $G_s = (N, L, N_r, L_r)$, an undirected graph ($G_s \geqslant G_V$). N denotes the set of substrate nodes and L denotes the set of substrate links. Each substrate network node $ns \in N$ is capable of providing resources for a VN vertex. N_r denotes the available amount of substrate resources on the substrate nodes to serve these VN vertices. Similarly, each substrate network link $es \in L$ is capable of providing resources for any VN edge. L_r denotes the available amount of substrate resources on the substrate links to serve these VN edges. The substrate network owned and managed by InP is denoted as G_s^{InP}. The multiple InPs inform the VN providers about their available substrate resources.

2.3 VN provisioning

It is the combined task of the VN provider and the InPs to provision the VN requests. The responsibility of the InPs is to publish their available substrate network resources to the VN providers. As and when a VN request G_v arrives, the VN provider identifies the candidate substrate network managed by the InP i.e. G_s^{InP} to find an appropriate VN mapping for it. After this, the allocation of resources from the server to the VN vertices and bandwidth resources to the VN edges is carried out. The mapping of a virtual network onto the substrate network is a two-step process: (a) VN vertex mapping and (b) VN edge mapping. In the VN vertex mapping step, the VN vertices from a VN request are managed by different substrate nodes as their hosts. In the VN edge mapping step, each virtual edge is mapped onto a set of substrate network links (substrate path) joining the corresponding substrate nodes. The substrate nodes, that host a VN vertex, have to provide a particular amount of the substrate resources to run the

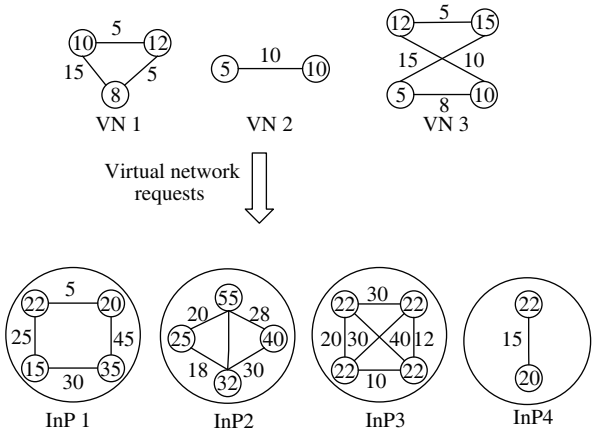

Figure 1 VN requests embedded on substrate networks owned by multiple infrastructure providers.

application of the user. Similarly, the substrate links reserve bandwidth resources for the corresponding VN edges for the data and information exchange among the VN vertices.

More formally, the virtual network embedding problem is described as follows: Given a number of substrate network graphs $G_s = (N, L, N_r, L_r)$ owned by multiple InPs and a number of virtual network requests $G_V = (V, E, V_r, E_r)$, the problem is how to choose a candidate InP and allocate its substrate network resources among the virtual network requests such that the revenue of the InPs, acceptance ratio of the VNs, link and node utilization of the SNs etc. are optimized. Figure 1 depicts a scenario of VN requests and the substrate networks owned by multiple infrastructure providers which serve these requests. The CPU is quantified as a resource for nodes/vertices and the values inside them determine their weights. Similarly, bandwidth is quantified as a resource for links/edges and values on them determine their weights.

3 The proposed model

A GA based model for the problem stated in Section 2 is being proposed in this section. An overview of GA can be found in [18] though briefly introducing, GA is a nature-inspired search based algorithm for combinatorial optimization problem. To start with, a set of random potential solutions are generated which evolves over the generations. GA exploits and explores new search space using survival-of-the fittest technique. In each new generation, solutions that are poor are discarded while better solutions mate and produce more improved solutions. A simple genetic algorithm consists of an initial population followed by selection, crossover and mutation operations:

Selection. The act of selecting good population among the chromosomes through some objective function (fitness function) used to rank the quality of the chromosomes.

Crossover. The act of swapping gene values between a pair of potential chromosomes to obtain the new chromosome, while simulating the mating of these two solutions.

Mutation. The act of randomly altering the value of a gene in a potential chromosome.

For the proposed GA based VNE model, various modules are as follows.

3.1 The problem structure

Proposed model is based on GA and various modules related with GA based VNE problem are discussed in this section.

3.1.1 *Chromosome structure*

Chromosome, in a GA based approach, is the potential solution to the problem that is scored for its fitness towards a given objective. Normally, the chromosome consists of alleles which in turn made of

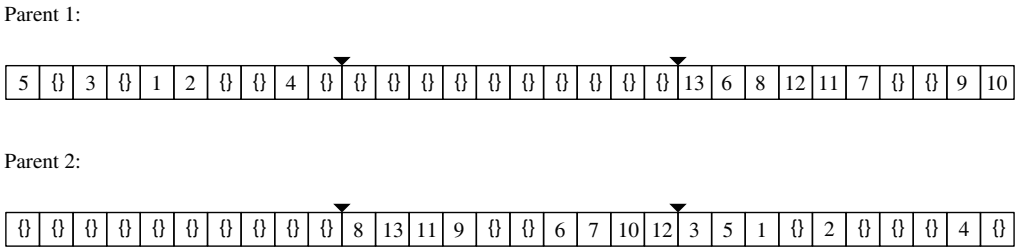

Figure 2 A chromosome encoding.

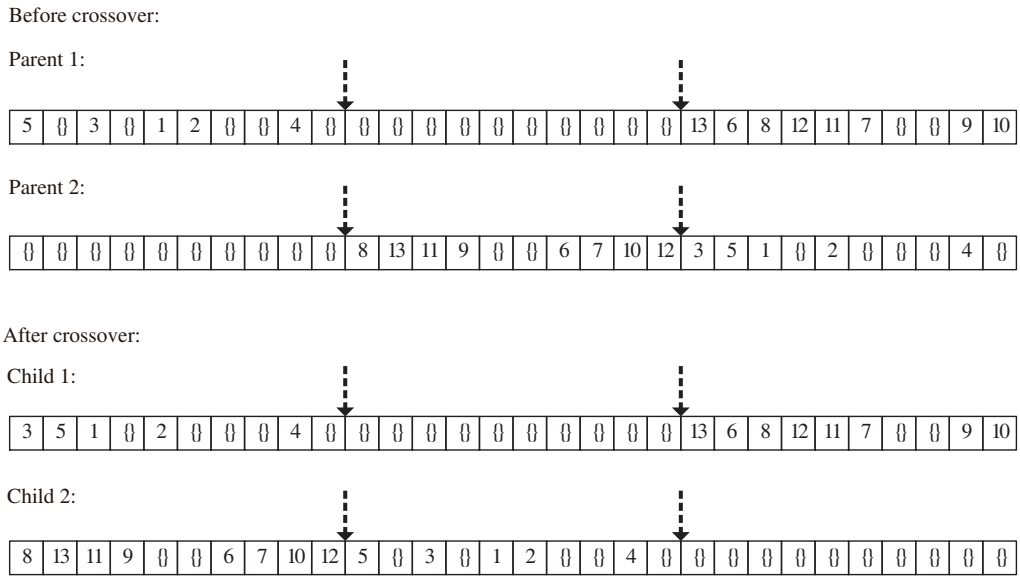

Figure 3 Multipoint crossover operation.

various genes that characterize the quality of the solution. In the proposed VNE problem, the solution is encoded using integer encoding as with binary encoding it would become too large to incorporate the required combined information of the virtual networks and substrate networks. The chromosome string representation is described through an example in Figure 2.

Two chromosomes, represented in Figure 2, are generated randomly. The total number of alleles in the chromosome is equal to the number of substrate networks available to serve the multiple virtual network requests. The size of each allele in a chromosome is kept equal to the size of the largest of all the substrate networks serving the VN requests i.e. the one with the maximum number of nodes. The values in each allele are filled again in a random manner with the VN vertices of the VN request filling the random positions of an allele, representing a particular substrate network. For example in parent 1, first position of the first allele is filled with value 5, third position with value 3, fifth position with value 1 and so on. This means that the vertex 5 of the virtual network is mapped to the 1st node of the substrate network, vertex 3 to the 3rd node of the substrate network, vertex 1 to the 5th node and so on. The allele that has no values assigned to it and is empty represents that no virtual network is mapped on this particular substrate network represented by that allele.

3.1.2 *Crossover*

After the random initial population generation, crossover is performed on the parent chromosomes. Multipoint crossover [19] is used for the purpose and the crossover points are chosen such that it represents the beginning of the new allele of the substrate network. These multiple points are denoted by arrows in Figure 3 where the crossover operation is demonstrated.

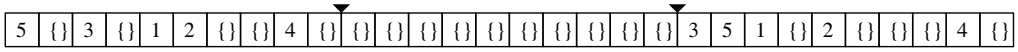

Figure 4 An infeasible chromosome.

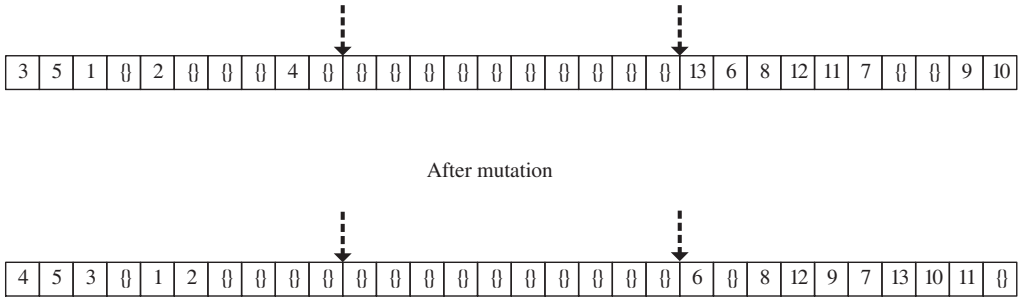

Figure 5 Mutation operation.

Table 1 Fitness function formulation notations

$\mathrm{SN} = \mathrm{SN}_1, \mathrm{SN}_2, \mathrm{SN}_3 \ldots$	Set of substrate networks owned by multiple InPs
$\mathrm{VN} = \mathrm{VN}_1, \mathrm{VN}_2, \mathrm{VN}_3 \ldots$	Set of virtual network requests to be mapped on multiple InPs SN
$\mathrm{VN}_i = V_{i1}, V_{i2}, V_{i3} \ldots$	Set of vertices in the ith VN
A_{ij}	jth vertex of the ith VN
P_{ij}	Position of A_{ij} i.e. SN node to which A_{ij} is mapped
$U[A_{ij}]$	Weight of A_{ij} i.e. resource (CPU) requested by jth vertex of the ith VN
$V[P_{ij}]$	Weight of P_{ij} i.e. available resource (CPU) on P_{ij}
$E_{A_{ijk}}$	Edge connecting A_j vertex to A_k vertex of ith VN
$E_{P_{ijk}}$	Smallest link (path with minimum weight) connecting P_{ij} node to P_{ik} node
$w(E_{A_{ijk}})$	Weight on the edge $E_{A_{ijk}}$ i.e. bandwidth requested by virtual network edge $E_{A_{ijk}}$
$w(E_{P_{ijk}})$	Weight on the link $E_{P_{ijk}}$ i.e. bandwidth available on substrate network link $E_{P_{ijk}}$

3.1.3 *Feasibility check*

It has been observed that after crossover, few of the solutions (children) generated are invalid. Therefore, a feasibility test is performed on the offspring. The solutions in which one virtual network is seen to be mapped on two substrate networks are infeasible solutions hence discarded. The crossover operation is repeated again to get feasible solutions. Child 1 and child 2 in Figure 3 are examples of feasible solutions. Figure 4 shows an infeasible chromosome that is to be discarded. This is because the same virtual network with 5 nodes is mapped on substrate network 1 as well as substrate network 3 making it an infeasible solution.

3.1.4 *Mutation*

After crossover, the newly generated offspring undergoes mutation (Figure 5). Mutation is performed so that GA overcomes a local optimum and also introduces new genes in to the population. In the mutation phase, the individual elements of every allele of a child are shuffled with a mutation probability discussed further in the performance evaluation section.

3.1.5 *Fitness function*

Residual resources of the substrate networks is an important parameter for most of the VNE models [20, 21]. The proposed work also considers residual resources of the substrate networks as a parameter to be optimized. The solution results in proper mapping of the VN requests while maximizing the revenue of the multiple InPs. To derive this fitness function, Table 1 defines the notations of different parameters used.

Weight difference W_{ij} of VN vertices A_{ij} and the substrate network nodes P_{ij} is calculated as

$$W_{ij} = V[P_{ij}] - U[A_{ij}]. \tag{1}$$

Node weight difference (Node_Weigh_diff) for all the corresponding virtual networks and substrate network pairs represented in an individual chromosome is given as

$$\text{Node\_Weigh\_diff} = \sum_i \sum_j W_{ij}. \tag{2}$$

Define a set, $S^i = \{(A_{ij}, P_{ij}) | \forall\ j \in \text{VN}_i\}$ (obtaining VN vertices and the corresponding SN node pairs for every VN request mapped on SN).

Edge_Weigh_diff is obtained by summing up all the corresponding edge/link weight differences of the virtual networks mapped on substrate networks for an individual chromosome as

$$\text{Edge\_Weigh\_diff} = \sum_i \sum_{j,k} (w(E_{P_{ijk}}) - w(E_{A_{ijk}})) \tag{3}$$

for each unique $\langle (A_{ij}, P_{ij}), (A_{ik}, P_{ik}) \rangle \in$ ordered pair of S^i. Fitness Z for an individual chromosome is calculated by adding the node and edge weight differences as

$$Z = \text{Node\_Weigh\_diff} + \text{Edge\_Weigh\_diff}. \tag{4}$$

After the fitness for all the chromosomes is calculated, the chromosome that has the minimum fitness value is selected as

$$\text{Weights\_diff} = \min_{k=1}^{\text{popsize}} (Z). \tag{5}$$

Here, popsize is the total population size.

This minimum summation of vertex-node resource difference and corresponding edge-link resource difference helps in finding the near optimal mapping of the VN and hence the best suitable InP. It results in saving of maximum amount of resources of the substrate networks which can be used to further embed other VNs as the best fit InP is selected for mapping VN request by calculating minimum weight difference. Thus, when the population converges and the final embedding is achieved with minimal unutilized substrate network resources, the revenue of InPs tend to increase as more substrate networks resources are free to embed other incoming VN requests. Consequently, the acceptance rate of the VN request increases which follows with the efficient utilization of the overall underlying substrate networks.

3.1.6 *Selection*

Selection is carried out to select the parent chromosomes for the reproduction of the offspring. The selected chromosomes are then assigned opportunities to reproduce. Several types of parent selection methods exist such as roulette wheel selection, random selection, rank selection, elitism selection, tournament selection etc. In this work, the selection is done through sorting wherein the best half chromosomes with respect to the fitness score are selected as the parent chromosomes.

3.2 Algorithm

Algorithm 1 is the pseudo-code of the algorithm, for the proposed model.

4 Performance evaluation

The simulation is done in MATLAB to visualize the performance of the proposed model. Experimentation is done ten times on various parameters and the average result is shown. This is to ensure the validity of the results produced. The comparison of the proposed GA based VNE approach in multiple InP scenario is carried out in two sections: (1) with other meta-heuristic VNE strategies and (2) with other multiple

Algorithm 1 GA based VNE algorithm

```
Generate initial population P1 randomly // random VN mapping chromosomes are generated
Calculate the fitness of each chromosome using the fitness function
for i = 1; i ⩽ number_of_generations; i++ do
    Sort the population of P1
    Select the best half of P1 and store it in a new population P2
    Randomly select pairs of parents from the population P2
    Perform crossover based on the crossover probability on the parents to produce offspring
    Apply feasibility check on each new individual
    Mutate offspring based on the mutation probability
    Store this newly generated population in P1
    Evaluate the fitness of each new individual in P1
end for
```

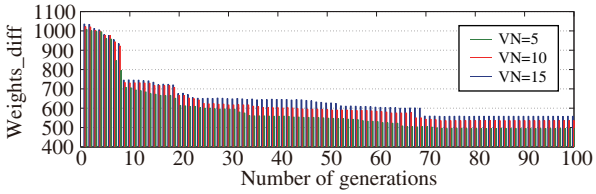

Figure 6 (Color online) Fitness for small sized VNs.

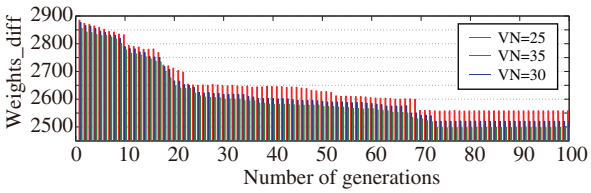

Figure 7 (Color online) Fitness for medium sized VNs.

InP based virtual network embedding strategies present in the literature. The input parameters, for the experiments, are similar to as described in [2,6,7] and are as follows.

- Population size is 50 (generated randomly).
- Size of substrate network varies uniformly in a range of 1 to 50 nodes.
- Numbers of virtual nodes in virtual networks are uniformly distributed between 2 and 10.
- All pairs of substrate nodes and VN vertices are randomly connected with a probability of 0.5
- Weights on the nodes and links of the SN are uniformly distributed between 50 and 100.
- Weights on the vertices of the VN requests are uniformly distributed within a range of 0 to 20
- Weights on the VNs edges are uniformly distributed between 0 and 50.
- The arrival of VN requests is modeled by a Poisson process with rate $\lambda_A = 4$ VN requests per 100 time unit
- VN lifetime is modeled by exponential distribution with mean $\mu_L = 100$ time units.
- Crossover probability is 0.7 and the mutation probability is 0.03.

4.1 Observation on weight difference

Experiment 1: Small sized VNs. Small virtual network sets comprise of VN vertices varying from 2 to 15. The substrate networks to map these VN requests vary in a range of 20 to 30 in number. Other input parameters are as given above. GA is iterated for 100 generations.

The fitness function is evaluated on three random instances of the varying VN size i.e. 15, 10 and 5 and the result is shown in Figure 6.

It can be concluded from Figure 6 that the average weights_diff for VN = 5 is 582.32, for VN = 10 is 626.06, and for VN = 15 is 649.29 which obviously relates to the fact that smaller sized VNs with small weights on vertices and edges, when mapped on the substrate networks managed by the InPs result in lower weights difference as compared to large sized VNs.

Experiment 2: Medium sized VNs. Medium virtual network sets comprise of VN vertices varying from 15 to 35.The substrate networks, to map these VN requests, vary in a range of 30 to 40 in number. Other input parameters are same as given above and GA is iterated for 100 generations.

The fitness function is evaluated on three random instances of the varying VN size i.e. 35, 30 and 25 and the result is shown in Figure 7.

It can be concluded from Figure 7 that the average weights_diff for VN = 25 is 2595.62, for VN = 30 is 2614.62, and for VN = 35 is 2644.85 which obviously relates to the fact that smaller sized VNs with

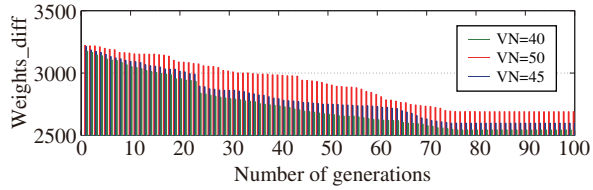

Figure 8 (Color online) Fitness for large sized VNs.

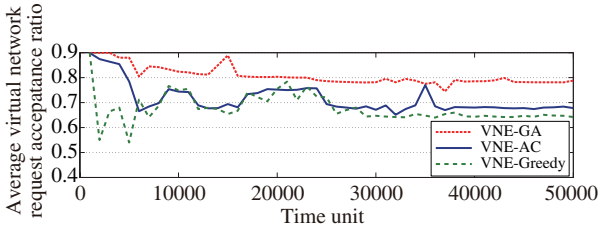

Figure 9 (Color online) Average VN acceptance ratio.

small weights on vertices and edges, when mapped on the substrate networks managed by the InPs result in lower weights difference as compared to large sized VNs.

Experiment 3: Large sized VNs. Large virtual network sets comprise of VN vertices varying from 35 to 50. The substrate networks to map these VN requests vary in a range of 40 to 50 in number. Other input parameters remain same as given above and GA is iterated for 100 generations.

The fitness function is evaluated on three random instances of the varying VN size i.e. 50, 45 and 40 and the result is shown in Figure 8.

It can be concluded from Figure 8 that the average weights_diff for VN = 40 is 2731.49, for VN = 45 is 2792.83, and for VN = 50 is 2905.41 which obviously relates to the fact that smaller sized VNs with small weights on vertices and edges, when mapped on the substrate networks managed by the InPs result in lower weights difference as compared to large sized VNs.

The observations derived from Figures 6–8 for all sized VN embedding is as follows.

(i) The large sized VNs have higher weights_diff as compared to small and medium sized VNs.

(ii) The virtual network requests with large number of vertices and favorably highly weighted vertices and edges have greater weights_diff than those VN requests which have less number of vertices and low weights on the vertices and edges.

(iii) 100 generations are sufficient to converge GA in all the experiments.

4.2 Comparison with other meta-heuristic based VNE strategies

Experiment 1: Observation on acceptance ratio of VN requests. The fitness function is minimized using GA which resulted in the acceptance of more and more VN requests to be embedded on the substrate networks. Acceptance ratio metric measures the percentage of accepted virtual network requests by an algorithm over a given period of time. The next set of experiment compares the acceptance ratio of VNs of the proposed GA based VNE algorithm with existing embedding strategies VNE-AC and VNE-Greedy. The acceptance ratio, in (6), is calculated as given in [8]. Simulation is executed for 50000 time units and is depicted through a graph in Figure 9.

$$\text{Accepted\_VN} = \frac{n(\text{VN}')}{n(\text{VN})} = \frac{(\text{number of VN requests accepted})}{(\text{number of VN requests})}. \tag{6}$$

Figure 9 shows the comparative analysis of the proposed VNE-GA with two other virtual network embedding strategies, VNE-AC [16] and VNE-Greedy [6]. The figure notably shows that the acceptance ratio of the VN requests by the proposed VNE-GA is higher compared to VNE-AC and VNE-Greedy. The VNE-Greedy algorithm does not consider the congested substrate network links which results in the decrement of the number of VN requests to be accepted for mapping. Moreover, the mapping of virtual vertices and edges is not coordinated in this approach. Also for VNE-AC approach, it is conspicuous that the acceptance ratio peak is lower than that of VNE-GA acceptance ratio peak. The reason for this is that the proposed approach tends to maximize comparatively more and more residual resources of the substrate network.

Experiment 2: Observation on average revenue. Average revenue is evaluated by calculating the overall benefit of all the accepted virtual network requests during a given time period. It measures the gain earned by the substrate networks provider and is expressed as in (7) similar to [3, 6, 22]. While minimizing the fitness function, the average revenue earned through the proposed VNE-GA algorithm is

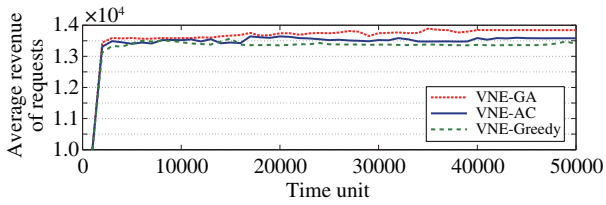

Figure 10 (Color online) Average revenue of VN requests.

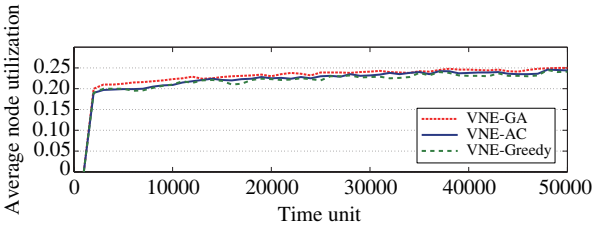

Figure 11 (Color online) Average SN node utilization.

compared with VNE-AC and VNE-Greedy approach. Experiment is performed for 50000 time units and the result is depicted by a graph in Figure 10.

$$R(G_v(t)) = \sum_{i=0}^{i=\text{All\_VN}} \alpha_i \left(\sum_{\text{ev}\in E} \mu \times E_r(\text{ev}) + \sum_{\text{nv}\in V} \lambda \times V_r(\text{nv}) \right), \tag{7}$$

where $R(G_v(t))$ is the revenue of serving the VN requests sent by the SPs at time t, All_VN represents all the virtual network requests, $\alpha_i \in 0, 1$ and $\alpha_i = 1$ if the ith VN request sent by the SP is fulfilled successfully. μ represents the unit revenue of bandwidth and λ represents the unit revenue of CPU.

Figure 10 shows consistently better performance of VNE-GA compared to VNE-AC [16] and VNE-Greedy [6] as higher revenue is earned by the InPs managing substrate networks, with this approach. As the acceptance ratio of the VN requests is directly proportional to the revenue earned by the InPs, the larger number of accepted VN requests seen in Figure 9, leads to the increase of the SN resources allocated to the VNs and hence higher revenue for the InPs is obtained.

Experiment 3: Observation on node utilization. As discussed earlier, since VNE-GA has the highest acceptance ratio, it also shows the highest growth in node utilization. The average node utilization of the substrate network is measured by averaging the stress of all the substrate nodes of the substrate networks. The substrate nodes stress is calculated using (8) as given in [22] and is defined as the total amount of CPU capacity allocated to different virtual nodes hosted on the substrate node ns $\in N$. The comparison on this performance metric for the proposed VNE-GA algorithm is done again with VNE-AC and VNE-Greedy approach. Simulation is executed for 50000 time units and the result is depicted by a graph in Figure 11.

$$\text{Stress(ns)} = \sum_{\text{nv}\to\text{ns}} V_r(\text{nv}). \tag{8}$$

Figure 11 depicts the average SN node utilization for different VN embedding algorithms. It is obvious from the graph that VNE-GA increases the node utilization. This is a result of higher acceptance ratio of VN requests by using VNE-GA.

Experiment 4: Observation on link utilization. The proposed VNE-GA algorithm achieves a highly optimized path to map the virtual network edges and thus leads to lower link utilization. The average link utilization of the substrate network is measured by averaging the stress of all the substrate links of the substrate networks. The stress on substrate links is calculated using (9) as given in [22] and is defined as the total amount of bandwidth reserved for the virtual links whose substrate paths pass through the substrate link es $\in L$. The comparison on this performance metric for the proposed VNE-GA algorithm is done again with VNE-AC and VNE-Greedy approach. Experiment is executed for 50000 time units and is depicted by a graph in Figure 12.

$$\text{Stress(es)} = \sum_{\text{ev}\to\text{es}} E_r(\text{ev}). \tag{9}$$

Figure 12 shows the comparative analysis of the average link utilization for different virtual network embedding algorithms. With the minimization of the fitness function, VNE-GA also tends to decrease the link utilization. While the graph in Figure 12 shows only a minimal variation in the curves of all the three algorithms, still VNE-GA depicts lower average link utilization.

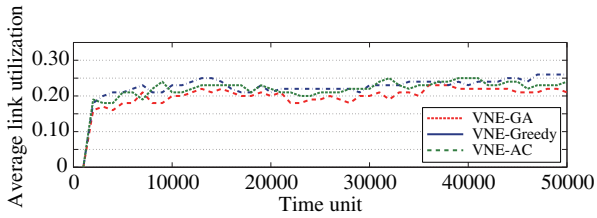

Figure 12 (Color online) Average link utilization.

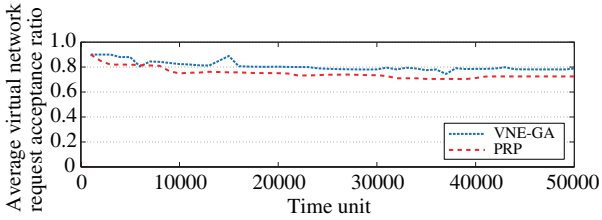

Figure 13 (Color online) Average virtual network request acceptance ratio.

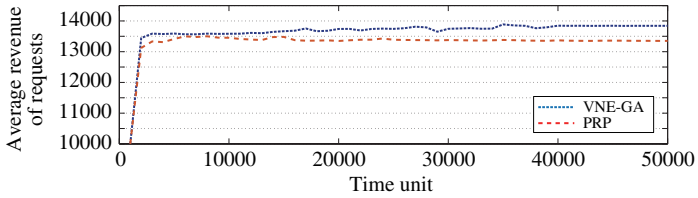

Figure 14 (Color online) Average revenue of VN requests.

4.3 Comparison with contemporary multiple InP based VNE approach

To the best of our knowledge, not much work is done on multiple InP based virtual network embedding in the literature [23,24]. Thus, the proposed approach is compared to [24] in terms of request acceptance ratio and revenue. VNE-GA is more quick to react to dynamically arriving VN requests unlike the compared exact parallel request processing (PRP) approach and thus acceptance rate of VN requests using VNE-GA is high. Also, PRP is only efficient for small/medium scale networks whereas VNE-GA embedding solution is suitable for large scale networks.

Experiment 1: Observation on acceptance ratio of VN requests. The proposed embedding algorithm VNE-GA provides higher acceptance ratio compared to the existing approach of exact mapping with PRP [24].

Figure 13 confirms that the VNE-GA embedding algorithm provides higher acceptance ratio when compared to the exact PRP approach. This result proves that the fitness function formulated in the proposed algorithm aids in saving large amount of substrate network resources. This ensures better use of substrate resources, thus increasing acceptance ratio compared to other approaches.

Experiment 2: Observation on average revenue. The revenue earned by multiple InPs also increases with the acceptance of more and more VN requests. The proposed VNE-GA algorithm shows an edge over PRP in this regard.

Figure 14 depicts the advantages of adopting VNE-GA approach compared to the PRP approach in terms of revenue. The vertical gap between the revenue curves evaluates the gain in revenue earned using the VNE-GA approach over the PRP approach. This gain results from the efficient utilization of substrate network resources managed by multiple InPs during the embedding in VNE-GA algorithm.

5 Conclusion

The work in this paper, addresses the problem of optimal provisioning of multiple virtual network requests among multiple infrastructure providers. A GA based virtual network embedding algorithm is proposed to address this NP - class problem. The model has been simulated and performance study is done on various parameters such as revenue generation for InPs, acceptance ratio and efficient node and link utilization. The efficacy of the proposed model is validated through the experimental results. The comparative study of the proposed models with other contemporary models exhibit that on the mentioned characteristic parameters the proposed model performs consistently well. Future work will focus on post-fault tolerant virtual network embedding strategies to be incorporated in the virtual network embedding model.

Acknowledgements This work was supported by UGC-UPEII, New Delhi. Also, authors accord their sincere thanks to the anonymous reviewers for the useful suggestions.

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Anderson T, Peterson L, Shenker S, et al. Overcoming the Internet impasse through virtualization. Computer, 2005, 38: 34–41
2 Zhu Y, Ammar M H. Algorithms for assigning substrate network resources to virtual network components. In: Proceedings of the 25th IEEE International Conference on Computer Communications, Barcelona, 2006. 1–12
3 Lischka J, Karl H. A virtual network mapping algorithm based on subgraph isomorphism detection. In: Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures, Barcelona, 2009. 81–88
4 Lu J, Turner J. Efficient mapping of virtual networks onto a shared substrate. Technical Report WUCSE-2006-35. Washington University, 2006
5 Houidi I, Louati W, Zeghlache D. A distributed virtual network mapping algorithm. In: Proceedings of IEEE International Conference on Communications, Beijing, 2008. 5634–5640
6 Yu M, Yi Y, Rexford J, et al. Rethinking virtual network embedding: substrate support for path splitting and migration. ACM SIGCOMM Comput Commun Rev, 2008, 38: 17–29
7 Chowdhury N M M K, Rahman M R, Boutaba R. Virtual network embedding with coordinated node and link mapping. In: Proceedings of the 28th IEEE International Conference on Computer Communications, Rio de Janeiro, 2009. 783–791
8 Melo M, Sargento S, Killat U, et al. Optimal virtual network embedding: Node-link formulation. IEEE Trans Netw Serv Manag, 2013, 10: 356–368
9 Butt N F, Chowdhury M, Boutaba R. Topology-awareness and reoptimization mechanism for virtual network embedding. In: Proceedings of the 9th International IFIP TC 6 Networking Conference, Chennai, 2010. 27–39
10 Nogueira J, Melo M, Carapinha J, et al. Virtual network mapping into heterogeneous substrate networks. In: Proceedings of IEEE Symposium on Computers and Communications (ISCC). IEEE: Washington, DC, 2011. 438–444
11 Dorigo M, Caro G D, Gambardella L M. Ant algorithms for discrete optimization. Artif Life, 1999, 5: 137–172
12 Kennedy J. Particle swarm optimization. In: Sammut C, Webb G I, eds. Encyclopedia of Machine Learning. New York: Springer US, 2010. 760–766
13 Chu S-C, Tsai P-W. Computational intelligence based on the behavior of cats. Int J Innov Comput Inform Control, 2007, 3: 163–173
14 Golberg D E. Genetic Algorithms in Search, Optimization, and Machine Learning. Boston: Addison-Wesley Longman Publishing Co., Inc., 1989
15 Zhang Z B, Cheng X, Su S, et al. A unified enhanced particle swarm optimization-based virtual network embedding algorithm. Int J Commun Syst, 2013, 26: 1054–1073
16 Fajjari I, Aitsaadi N, Pujolle G, et al. VNE-AC: virtual network embedding algorithm based on ant colony metaheuristic. In: Proceedings of IEEE International Conference on Communications (ICC), Kyoto, 2011. 1–6
17 Mi X M, Chang X L, Liu J Q, et al. Embedding virtual infrastructure based on genetic algorithm. In: Proceedings of the 13th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), Beijing, 2012. 239–244
18 Sivanandam S N, Deepa S N. Introduction to Genetic Algorithms. Berlin/Heidelberg: Springer-Verlag, 2007
19 Spears W M, Jong K A D. An analysis of multi-point crossover. Technical Report, DTIC Document, 1990
20 Zhang S, Qian Z Z, Wu J, et al. An opportunistic resource sharing and topology-aware mapping framework for virtual networks. In: Proceedings of the 31st Annual IEEE International Conference on Computer Communications, Orlando, 2012. 2408–2416
21 Rahman M R, Boutaba R. SVNE: survivable virtual network embedding algorithms for network virtualization. IEEE Trans Netw Serv Manag, 2013, 10: 105–118
22 Chowdhury M, Rahman M R, Boutaba R. Vineyard: virtual network embedding algorithms with coordinated node and link mapping. IEEE/ACM Trans Netw, 2012, 20: 206–219
23 Hasan M M, Amarasinghe H, Karmouch A. Network virtualization: dealing with multiple infrastructure providers. In: Proceedings of IEEE International Conference on Communications (ICC), Ottawa, 2012. 5890–5895
24 Houidi I, Louati W, Ameur W B, et al. Virtual network provisioning across multiple substrate networks. Comput Netw, 2011, 55: 1011–1023