


RESEARCH ARTICLE

# An algorithm for fragment-aware virtual network reconfiguration

Xinbo Liu \*, Buhong Wang 

Information and Navigation college, Air Force Engineering University, Xi'an, Shaanxi, China

 These authors contributed equally to this work.

\* [zhongxing83@126.com](mailto:zhongxing83@126.com)

## Abstract

In view of the fact that the current online virtual network embedding algorithms do not consider the fragment resources generated in the embedding process deeply enough, resulting in the problem that the acceptance ratio and the revenue to cost ratio are both low, a mathematical model for virtual network reconfiguration is constructed and a heuristic algorithm for fragment-aware virtual network reconfiguration (FA-VNR) is proposed. The FA-VNR algorithm selects the set of virtual nodes to be migrated according to the fragment degrees of the physical nodes, and selects the best virtual node migration scheme according to the reduction of the fragment degrees of the physical nodes as well as the reduction of the embedding cost of the embedded virtual networks. Extensive simulation results show that the proposed FA-VNR algorithm not only can obviously improve the acceptance ratio and the revenue to cost ratio of the current online virtual network embedding algorithm, but also has better optimization effect than the existing virtual network reconfiguration algorithm.



## OPEN ACCESS

**Citation:** Liu X, Wang B (2018) An algorithm for fragment-aware virtual network reconfiguration. PLoS ONE 13(11): e0207705. <https://doi.org/10.1371/journal.pone.0207705>

**Editor:** Halife Kodaz, University of Konya Technical, TURKEY

**Received:** May 29, 2018

**Accepted:** November 4, 2018

**Published:** November 21, 2018

**Copyright:** © 2018 Liu, Wang. This is an open access article distributed under the terms of the [Creative Commons Attribution License](https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

**Data Availability Statement:** All relevant data are within the paper and its Supporting Information files.

**Funding:** The author(s) received no specific funding for this work.

**Competing interests:** The authors have declared that no competing interests exist.

## Introduction

Network virtualization allows multiple heterogeneous virtual networks (VNs) to share and run on the same physical network, which can solve the ossification problem of Internet [1–4]. However, network virtualization brings flexibility to the network architecture while also presenting new challenges. One of the major challenges in network virtualization is how to efficiently and effectively allocate physical network resources to the arriving VN requests, which is known as the virtual network embedding (VNE) problem.

In order to achieve network virtualization as soon as possible, in recent years, the VNE problem has mostly been of academic interest, and a lot of online VNE algorithms have been proposed [5–9]. Authors in [5] propose a VNE algorithm named VNE-DCC, whose acceptance ratio and revenue to cost ratio are attractive. The VNE-DCC algorithm considers not only the local resources of a node, but also the topological attributes of its neighborhood nodes. However, the VNE-DCC algorithm does not consider how to reduce the fragment resources generated in the physical network during the VNE process.

For the online VNE algorithms, the arrival time, resource demand, and lifetime of each VN request are random variables. With the arrival and departure of the VN requests, the resources of the physical network may be fragmented [10], and the embedding schemes for the VN

requests that have been allocated resources on the physical network may become non-optimal [11]. These will cause the rejection of future VN requests. Therefore, in order to improve the performance of the online VNE algorithms, it is necessary to introduce the reconfiguration mechanism into the online VNE process, and dynamically adjust the node load and link load of the physical network.

Compared with online VNE problem, the VN reconfiguration problem is relatively less explored and only a few VN reconfiguration algorithms are proposed. Authors in [12] give a reactive VN reconfiguration scheme for the online VNE problem, where the VN reconfiguration algorithm is run only when a VN embedding strategy fails to admit a VN request. The main contribution of their work is the proposal of an integer linear programming (ILP) formulation for the reconfiguration. The simulation results show that the acceptance ratio can be significantly improved with reconfigurations. However, the ILP problem belongs to the NP-hard problem. When the instance of the problem is large, it is difficult to find the optimal solution within a limited time. In order to minimize the execution time of VN reconfiguration algorithms, some scholars use heuristic-based or metaheuristic-based solutions to solve the VN reconfiguration problem. Authors in [13] propose a heuristic VN reconfiguration algorithm, in which the physical link bandwidth utilization is driven to trigger virtual link migration. Authors in [14] propose a VN reconfiguration approach, which is validated in different network size and link density scenarios. The pivotal element of their approach is the virtual link migration problem, which is meant to find optimal reconfigurations of virtual links on the physical network in runtime. Authors in [15] propose a simulated annealing based metaheuristic algorithm to solve the VN reconfiguration problem. The objective of their work is to reduce the number of bottleneck physical links as well as the total bandwidth consumption of the embedded VNs. Although the above algorithms improve the acceptance ratio of VN requests, they all pay too much attention to the reconfiguration of virtual links, and not pay enough attention to the reconfiguration of virtual nodes. Author in [16] propose a reactive VN reconfiguration algorithm to improve acceptance ratio. The main idea of the VN reconfiguration is to reconfigure the current embeddings to obtain the physical network with as many low-stress physical nodes and links as possible. Authors in [17] propose a VN reconfiguration algorithm based on minimum cost. The algorithm improves the acceptance ratio of the VN requests and the load balancing degree of the physical network by migrating fewer virtual nodes and virtual links. Based on the literature [17], authors in [18] consider the close relationship between the virtual nodes in the virtual node migration process, and further improve the acceptance ratio of the VN requests.

In summary, the existing VN reconfiguration algorithms either emphasize too much on reconfiguring the bottleneck physical links, or overemphasize the reconfiguration of the bottleneck physical nodes. In this paper, in order to further improve the performance of the existing VNE algorithms, we consider the VN reconfiguration problem from two aspects: bottleneck nodes and bottleneck links. We first define a method to measure the fragment degrees of the physical nodes, and then propose an algorithm for fragment-aware virtual network reconfiguration (FA-VNR). The FA-VNR algorithm selects the virtual nodes to be migrated and the target physical nodes from two aspects: resource fragment degree and embedding cost. The simulation results show that the FA-VNR algorithm can improve the performance of the online VNE algorithm, and is superior to the existing VN reconfiguration algorithm.

## Network model and problem description

### Physical network

The physical network is modeled as a weighted undirected graph  $G^p = (N^p, E^p)$ , where  $N^p$  is the set of physical nodes,  $E^p$  is the set of physical links. For a given physical node  $n^p \in N^p$ , we take

$cpu(n^p)$  and  $loc(n^p)$  to denote the available CPU resource and location information of  $n^p$ . Similarly, for a given physical link  $e^p \in E^p$ , we take  $b(e^p)$  to denote the available bandwidth of  $e^p$ .

### Virtual network

For the online VNE problem, the VNs that need to be embedded are a series of VN requests that arrive in a time series. For the  $m^{th}$  arriving VN request, we use  $VNR_m = (G_m^v, t_a, t_e)$  to represent, where  $G_m^v$  is the virtual network topology, and  $t_a$  and  $t_e$  are the arrival time and end time of the VN request, respectively. The virtual network topology is also represented by a weighted undirected graph  $G_m^v = (N_m^v, E_m^v)$ , where  $N_m^v$  is the set of virtual nodes and  $E_m^v$  is the set of virtual links. For a given virtual node  $n^v \in N_m^v$ , we take  $cpu(n^v)$  and  $loc(n^v)$  to denote the required CPU resource and location information of  $n^v$ . For a given virtual link  $e^v \in E_m^v$ , we take  $b(e^v)$  to denote the required bandwidth of  $e^v$ . When the VN reconfiguration starts each time, the VN requests that have been embedded and have not left are denoted by  $\bar{G} = (\bar{N}, \bar{E})$ , where  $\bar{N}$  and  $\bar{E}$  respectively represent the set of embedded virtual nodes and the set of embedded virtual links.

### VN reconfiguration

VN reconfiguration is the remapping of the VNs that have been allocated resources on the physical network. Through proper reconfiguration, it can not only improve the ability of the physical network to accept the future VN requests, but also reduce the embedding cost of the embedded VNs. Fig 1 depicts a simple example of the VN reconfiguration. In the figure, the attribute of any node  $n$ , i.e.  $cpu(n)$ , is depicted in the rectangle next to the node and the number next to any link  $e$  represents the attribute of the link, i.e.  $b(e)$ . The VN reconfiguration procedure can be demonstrated in Fig 1: (I) At time  $t_0$ , the VN request has not arrived yet, and no VN is hosted on the physical network. (II) At time  $t_1$ , the VN request  $VNR_1$  has arrived, its node embedding scheme is  $\{a \rightarrow A, b \rightarrow B\}$ , and the link embedding scheme is  $\{(a,b) \rightarrow (A,B)\}$ . (III) At time  $t_2$ , the VN request  $VNR_2$  has arrived, its node embedding scheme is  $\{d \rightarrow A, c \rightarrow C\}$ , and the link embedding scheme is  $\{(d,c) \rightarrow (A,B,C)\}$ . (IV) At time  $t_3$ , the VN request  $VNR_1$  has left, and only the  $VNR_2$  is hosted on the physical network. (V) At time  $t_4$ , the VN request  $VNR_3$  has arrived. At this time, if the virtual node  $c$  in  $VNR_2$  is migrated from the physical node C to the physical node B, it can not only release 20 units of bandwidth resources, but also

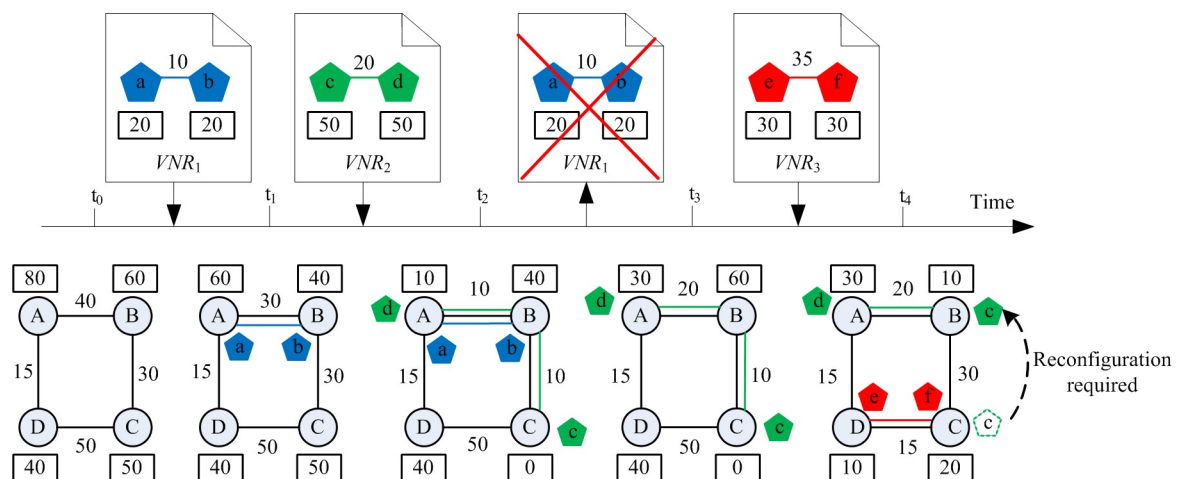


Fig 1. An example of VN reconfiguration.

<https://doi.org/10.1371/journal.pone.0207705.g001>

enable  $VNR_3$  to be embedded successfully; otherwise, if no reconfiguration operation is performed, the  $VNR_3$  will not be embedded successfully due to limited bandwidth requirement. Therefore, for the online VNE algorithm, selecting an appropriate VN reconfiguration algorithm can improve its performance.

### Reconfiguration objectives

The objective of VN reconfiguration is to further improve the performance of the existing VNE algorithms. Therefore, it is similar to the objectives of the existing VNE algorithm [6,19]. In this paper, the performance of VN reconfiguration algorithm is evaluated by using the acceptance ratio and revenue to cost ratio.

The acceptance ratio is defined as the ratio of the number of successfully embedded VN requests to the total number of VN requests within a certain period of time, as shown in Eq (1).

$$\eta(T) = \lim_{T \rightarrow \infty} \frac{\sum_{t=0}^T VNR_s}{\sum_{t=0}^T VNR_a} \tag{1}$$

Where  $VNR_s$  represents the VN requests which have been successfully embedded during the time from 0 to  $T$ ,  $VNR_a$  represents the VN requests which have arrived during the time from 0 to  $T$ .

The embedding revenue of  $VNR_m = (G_m^v, t_a, t_e)$  at time  $t$  is defined as Eq (2).

$$R(G_m^v, t) = \sum_{n^s \in N_m^v} cpu(n^s) + \sum_{e^{st} \in E_m^v} b(e^{st}) \tag{2}$$

The embedding cost of  $VNR_m = (G_m^v, t_a, t_e)$  at time  $t$  is defined as Eq (3).

$$C(G_m^v, t) = \sum_{n^s \in N_m^v} \sum_{n^i \in N^p} x_i^s \cdot cpu(n^s) + \sum_{e^{st} \in E_m^v} \sum_{e^{ij} \in E^p} y_{ij}^{st} \cdot b(e^{st}) \tag{3}$$

Where  $x_i^s$  is a binary variable denoting the embedding relation between the virtual node  $n^s$  and the physical node  $n^i$ , and  $y_{ij}^{st}$  is a binary variable denoting the embedding relation between the virtual link  $e^{st}$  and the physical link  $e^{ij}$ .  $x_i^s$  is 1 if the virtual node  $n^s$  is embedded onto the physical node  $n^i$ , and 0 otherwise.  $y_{ij}^{st}$  is 1 if the virtual link  $e^{st}$  is embedded onto the physical link  $e^{ij}$ , and 0 otherwise.

The revenue to cost ratio is defined as the ratio of the embedding revenue to the embedding cost in a certain period of time, as shown in Eq (4).

$$\varphi(T) = \lim_{T \rightarrow \infty} \frac{\sum_{t=0}^T \sum_{G_m^v \in VNR_s} R(G_m^v, t)}{\sum_{t=0}^T \sum_{G_m^v \in VNR_a} C(G_m^v, t)} \tag{4}$$

## Mathematical model of VN reconfiguration

### Definition of node’s fragment degree

Currently, typical VN reconfiguration algorithms [13–18], usually select virtual nodes (or virtual links) that need to be reconfigured only according to the load strength of the physical

nodes (or the physical links). However, these algorithms do not consider the node resources and their adjacent link resources together, and have some sidedness. Based on the existing researches, this paper defines the physical node fragment degree to better select the virtual nodes and their adjacent virtual links that need to be reconfigured.

In VNE, performance metrics such as latency, path length, and stress level are commonly considered to improve the quality of service (QoS) of the virtual networks; while to increase the economical profit of the physical network, it is necessary to consider the metrics such as acceptance ratio and revenue to cost ratio [10,20]. In this paper, the purpose of introducing the FA-VNR algorithm into VNE process is to further improve the economic benefits of the physical network. Therefore, when defining the fragment degree of a physical node, we only consider the CPU resource consumption of the physical node and the bandwidth resources consumption of its adjacent links. Consequently, for any physical node  $n^i$ , the definition of fragment degree is shown in Eq (5).

$$FD(n^i) = \alpha \cdot NS(n^i) + \beta \cdot ES(n^i) \tag{5}$$

Where  $NS(n^i)$  is the CPU resource load strength of  $n^i$ ,  $ES(n^i)$  is the adjacent link resource load strength of  $n^i$ ,  $\alpha$  is the weight coefficient of  $NS(n^i)$ , and  $\beta$  is the weight coefficient of  $ES(n^i)$ . The definitions of  $NS(n^i)$  and  $ES(n^i)$  are shown in Eq (6) and Eq (7), respectively.

$$NS(n^i) = \frac{\sum_{N_m^v \in \bar{N}} \sum_{n^s \in N_m^v} x_i^s \cdot cpu(n^s)}{cpu(n^i)} \tag{6}$$

$$ES(n^i) = \frac{\sum_{e^{ij} \in E(n^i)} \sum_{E_m^v \in \bar{E}} \sum_{e^{st} \in E_m^v} \gamma_{ij}^{st} \cdot b(e^{st})}{\sum_{e^{ij} \in E(n^i)} b(e^{ij})} \tag{7}$$

In Eq (7),  $E(n^i)$  represents the adjacent link set of physical node  $n^i$ .

### ILP formulation

For the VN reconfiguration problem, on the one hand, the fragmented resources in the physical network should be reduced as much as possible to improve the acceptance ratio of VN requests. On the other hand, in order to increase the revenue to cost ratio of the physical network, the embedding cost of the VNs that have been embedded should be reduced as much as possible. Therefore, in this paper, the following multi-objective integer linear programming formulation for VN reconfiguration is constructed.

Objectives:

$$\min \sum_{n^i \in NP} f(n^i) \tag{8}$$

$$\max \sum_{G_m^v \in \bar{G}} [C(G_m^v, rb) - C(G_m^v, ra)] \tag{9}$$

Constraints:

$$\forall n^i \in N^p, \sum_{N_m^v \in \bar{N}} \sum_{n^s \in N_m^v} x_i^s \cdot cpu(n^s) \leq cpu(n^i) \tag{10}$$

$$\forall N_m^v \in \bar{N}, \forall n^s \in N_m^v, \forall n^i \in N^p, x_i^s \cdot \text{dis}(\text{loc}(n^s), \text{loc}(n^i)) \leq D(n^s) \quad (11)$$

$$\forall N_m^v \in \bar{N}, \forall n^i \in N^p, \sum_{n^s \in N_m^v} x_i^s \leq 1 \quad (12)$$

$$\forall N_m^v \in \bar{N}, \forall n^s \in N_m^v, \sum_{n^i \in N^p} x_i^s = 1 \quad (13)$$

$$\forall e^{ij} \in E^p, \sum_{E_m^v \in \bar{E}} \sum_{e^{st} \in E_m^v} y_{ij}^{st} \cdot b(e^{st}) \leq b(e^{ij}) \quad (14)$$

$$\forall E_m^v \in \bar{E}, \forall e^{st} \in E_m^v, \forall n^i \in N^p, \sum_{n^j \in N^p} y_{ij}^{st} - \sum_{n^j \in N^p} y_{ji}^{st} = x_i^s - x_i^t \quad (15)$$

$$\forall N_m^v \in \bar{N}, \forall n^s \in N_m^v, \forall n^i \in N^p, x_i^s \in \{0, 1\} \quad (16)$$

$$\forall E_m^v \in \bar{E}, \forall e^{st} \in E_m^v, \forall e^{ij} \in E^p, y_{ij}^{st} \in \{0, 1\} \quad (17)$$

In Eq (8), when the fragment degree of the physical node  $n^i$  is greater than the threshold value  $FD_{th}$ , the value of  $f(n^i)$  is 1; otherwise it is 0. In Eq (9),  $C(G_m^v, rb)$  represents the embedding cost of the virtual network  $G_m^v$  before reconfiguration, and  $C(G_m^v, ra)$  represents the embedding cost of the virtual network  $G_m^v$  after reconfiguration. The Eq (10) indicates that the available CPU resources of the physical node should not be less than the CPU resource requirements of the virtual nodes it hosts. The Eq (11) indicates that the Euclidean distance between the physical node and its hosted virtual node should meet the requirements of the virtual node for the distance. The Eq (12) indicates that any physical node can only host at most one virtual node in the same VN request. The Eq (13) indicates that any virtual node can only be embedded onto one physical node. The Eq (14) indicates that the available bandwidth of any physical link should not be less than the bandwidth requirement of the virtual links it hosts. The Eq (15) indicates that any virtual link should be embedded onto a valid physical path. The Eqs (16) and (17) represent the range of values for variables  $x_i^s$  and  $y_{ij}^{st}$ .

### FA-VNR algorithm design

Due to the fact that multi-objective integer linear programming is NP-hard, obtaining the optimal solution of VN reconfiguration problem is NP-hard. When the problem size is small, tools such as GLPK and CPLEX [21,22] can be used for exact solution. When the problem size is large, it is difficult to find the optimal solution within a limited time. To solve this problem, a heuristic algorithm named FA-VNR is designed in this section. It answers the following key questions: (1) When does it need to perform VN reconfiguration? (2) Which virtual nodes and their adjacent links need to be migrated? (3) Where should the virtual nodes and their adjacent links be migrated? (4) How to carry out the migration? The main steps of the FA-VNR algorithm are as follows:

**Step 1:** Determine whether the VN reconfiguration algorithm need to be started. For VN reconfiguration algorithms, the choice of trigger condition is very important. When the VN reconfiguration algorithm triggers too frequently, it will take a lot of reconfiguration time. On the other hand, when the VN reconfiguration algorithm triggers too rarely, the fragment

resources and bottleneck resources in the physical network cannot be well reduced. In this paper, a simple and practical periodic VN reconfiguration mechanism is adopted, that is, a VN reconfiguration is performed at a certain interval of time  $T_r$ .

**Step 2:** Construct the set of physical nodes  $S^p$  that need to be reconfigured. For online VNE, the remaining CPU resources and link resources in physical network are dynamically changing over time. When the remaining resources of the physical network are less, the fragment degrees of all nodes are relatively large; when there are many remaining resources in the physical network, the fragment degrees of all nodes are relatively small. Therefore, this paper determines the set of physical nodes that need to be reconfigured by selecting a dynamic node fragment threshold. The node fragment threshold is defined as shown in Eq (18).

$$FD_{th} = (1 - \rho) \cdot \frac{\sum_{n^i \in N^p} FD(n^i)}{|N^p|} + \rho \cdot \max_{n^i \in N^p} FD(n^i) \tag{18}$$

Where  $\rho(0 \leq \rho \leq 1)$  is a proportional coefficient that is used to adjust the ratio of the mean value of the node fragment degree and the maximum value of the node fragment degree to the node fragment threshold. In this paper,  $\rho = 0.1$  is set in the simulation process.

Select the physical nodes with fragment degrees larger than the threshold value  $FD_{th}$  and store them in the physical node set  $S^p$ , in which the physical nodes need to be reconfigured.

**Step 3:** Construct the set of virtual nodes  $S^v$  that need to be migrated. When migrating a virtual node, not only the virtual node but also its adjacent virtual links must be migrated. Therefore, when selecting a virtual node that needs to be embedded, it is necessary to consider minimizing the fragment degree of the physical node that host the virtual node on the one hand, and on the other hand, it is necessary to consider minimizing the embedding cost of the virtual node and its adjacent virtual links. For this purpose, for virtual node  $n^v$ , we define the cost to revenue ratio of the virtual node and its adjacent virtual links as shown in Eq (19).

$$CR(n^v) = \frac{\sum_{e^v \in E(n^v)} hops(e^v) \cdot b(e^v)}{\sum_{e^v \in E(n^v)} b(e^v)} \tag{19}$$

Where  $E(n^v)$  represents the set of adjacent virtual links of the virtual node  $n^v$ , and  $hops(e^v)$  represents the hops of the physical path which host the virtual link  $e^v$ .

The process of selecting virtual nodes hosted on each physical node in  $S^p$  is divided into the following two steps: the first step is to sort the virtual nodes by the  $CR(n^v)$  values in descending order; the second step is to construct the set of virtual nodes to be migrated with the principle of migrating as few virtual nodes as possible. Based on above, the pseudo code of the algorithm for constructing the set of virtual nodes to be migrated is given by **Algorithm 1**.

**Algorithm 1** The algorithm for constructing the set of virtual nodes to be migrated

**Input:** Physical node set  $S^p$

**Output:** Virtual node set  $S^v$

1. **for**  $k = 1: |S^p|$  **do**
2.     Calculate  $CR(n^v)$  values of the virtual nodes that are hosted by the  $k^{th}$  physical node  $S_k^p$  in  $S^p$
3.     Sort the virtual nodes that are hosted by  $S_k^p$  in descending order of  $CR(n^v)$ , and record the result into  $VNL$
4.     **for**  $i = 1: |VNL|$  **do**
5.         Calculate  $FD(S_k^p)$  of  $S_k^p$  after migrating the first  $i$  virtual nodes in the  $VNL$

```

6.         if  $FD(S_k^v) \leq FD_{th}$  then
7.             Save the first  $i$  virtual nodes in  $VNL$  into  $S^v$ 
8.             break
9.         end if
10.    end for
11. end for
12. return  $S^v$ 

```

**Step 4:** Sort the virtual nodes in  $S^v$ . Considering that migrating a virtual node with a large  $CR(n^v)$  value is preferred, more physical network resources can be released, which is conducive to the migration of the subsequent virtual nodes. Therefore, in this step, the virtual nodes in  $S^v$  are arranged in descending order of  $CR(n^v)$  values and the virtual nodes with high  $CR(n^v)$  values are migrated preferentially.

**Step 5:** Construct the candidate target physical node set  $T^p$ . For each virtual node in  $S^v$ , the candidate target physical nodes are selected according to the following principles: (1) the target physical node cannot be a physical node that needs to be reconfigured; (2) any two virtual nodes in the same VN request cannot be embedded onto a same physical node; (3) the available CPU resource of the target physical node should be larger than the CPU resource requirement of the virtual node to be migrated; (4) the target physical node needs to satisfy the position constraint.

**Step 6:** Select the virtual node migration scheme. After the virtual node is migrated to the target physical node, the shortest path algorithm is used to re-embed the adjacent virtual links of the virtual node. When the virtual node  $S_k^v$  in the virtual network  $G_m^v$  is migrated from the physical node  $n^i$  to the physical node  $n^j$ , the amount of reduction in fragment degree and the amount of reduction in resource consumption of the physical network are shown in Eq (20) and Eq (21), respectively.

$$\delta_{FD} = [FD(n^j) + FD(n^i)] - [\overline{FD(n^j)} + \overline{FD(n^i)}] \tag{20}$$

$$\delta_C = C(G_m^v, rb) - C(G_m^v, ra) \tag{21}$$

In Eq (20),  $FD(n^j)$  represents the fragment degree of  $n^j$  before the migration of the virtual node;  $FD(n^i)$  represents the fragment degree of  $n^i$  before the migration of the virtual node;  $\overline{FD(n^j)}$  represents the fragment degree of  $n^j$  after the migration of the virtual node;  $\overline{FD(n^i)}$  represents the fragment degree of  $n^i$  after the migration of the virtual node.

When choosing the virtual node migration scheme, the fragment degree of the physical network should be reduced first and the resource cost of the physical network should be reduced as much as possible. Therefore, the following three principles are defined to determine the migration scheme of virtual node.

- i. The reduction of the fragment degree is greater than 0, that is,  $\delta_{FD} > 0$ ;
- ii. Minimize the consumption of physical network resources, that is,  $\delta_C \geq 0$ ;
- iii. The  $\delta_{FD}$  and  $\delta_C$  of all the migration schemes that satisfy the above conditions (I) and (II) are normalized, then the normalized  $\delta_{FD}$  and  $\delta_C$  are added to obtain  $\delta$ , and the migration scheme with the largest  $\delta$  value is selected as the final migration scheme.

Based on above, the pseudo code of the algorithm for selecting the virtual node migration scheme is given by **Algorithm 2**.

**Algorithm 2** The algorithm for selecting the virtual node migration scheme

**Input:** Virtual node  $S_k^v$ , the candidate physical node set  $T^p$

**Output:** The migration scheme of  $S_k^v$

```

1.   for  $i = 1:|T^p|$  do
2.       Migrate  $S_k^v$  to the  $i^{th}$  physical node  $T_i^p$  in  $T^p$ 
3.       if  $\delta_{FD} > 0$  &&  $\delta_C \geq 0$  do
4.           Save  $T_i^p$  into the available migration scheme set  $Can$ 
5.       end if
6.   end for
7.   if  $Can \subset \emptyset$  then
8.       return Virtual node  $S_k^v$  migration failed
9.   else
10.      for  $j = 1:|Can|$  do
11.          Normalize the  $\delta_{FD}$  and  $\delta_C$  of the  $j^{th}$  physical node in
12.           $Can$ , and add the two to get  $\delta$ 
13.      end for
14.      Select the physical node with the largest  $\delta$  value in  $Can$  as
15.      the migration scheme of  $S_k^v$ 
16.  end if
17.  return The migration scheme of  $S_k^v$ 

```

**Step 7:** Skip to step 5 and continue if the migration of all virtual nodes is not completed.

## Experimental environment setup and results analysis

### Experimental environment settings

In order to evaluate the performance of VNE algorithms, one has to perform appropriate simulations. The ALEVIN [23] is an excellent simulation tool for evaluating VNE algorithms. Unfortunately, it lacks the ability to evaluate online VNE approaches with reconfiguration [23]. Based on the simulation framework of ALEVIN, we developed a simulator using MATLAB to evaluate the performance of our online FA-VNR algorithms.

Similar to [6], we randomly generate the topologies of physical network and virtual networks by using GT-ITM tool [24,25]. We assume that the VN requests arrival in a Poisson process. The lifetime of each VN request follows an exponential distribution. The detailed parameters of the physical network and virtual networks used in our simulation are shown in Table 1.

In Table 1,  $[x_{min}, x_{max}]$  denotes a uniform distribution between  $x_{min}$  and  $x_{max}$ .  $Pois\{p\}$  and  $Expo\{e\}$  stand for the Poisson and exponential distributions with mean  $p$  and  $e$ , respectively.

In our simulation, we set the position constraint to  $D(n^s) = 40$ , the VN reconfiguration period to 500 time units, and test the performance of the FA-VNR algorithm from both the acceptance ratio and the revenue to cost ratio. The RTA-MAX algorithm proposed in [6] is used as the basic algorithm, and the FA-VNR algorithm proposed in this paper and the TA-VNR algorithm proposed in [18] are used to reconfigure the embedded VNs periodically. The detail of the compared algorithms is list in Table 2.

**Table 1. Simulation assumptions and parameters.**

Parameters	Physical network	Virtual networks
Number of nodes	100	[2, 10]
Node CPU resource	[50, 100]	[0, $M_{cpu}$ ]
Node position	Distributed in the scope $L \times L = 100 \times 100$	Distributed in the scope $L \times L = 100 \times 100$
Number of Links	500	Each pair of nodes is interconnected with a probability of 0.5
Link bandwidth resource	[50, 100]	[0, $M_{bw}$ ]
VN request arrival rate	-	$Pois\{0.05\}$
VN request life time	-	$Expo\{1000\}$

<https://doi.org/10.1371/journal.pone.0207705.t001>

**Table 2. Algorithm comparison.**

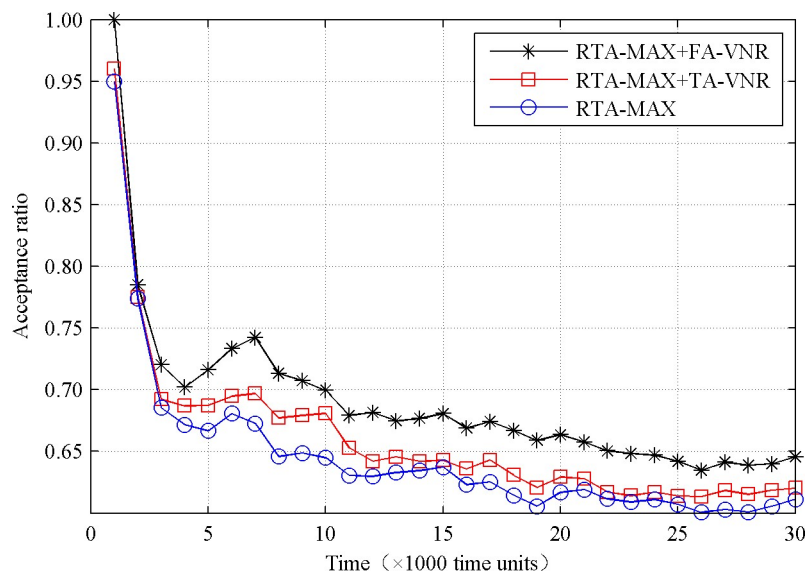
Notation	Description
RTA-MAX	It means that only the RTA-MAX algorithm is used in the VNE process and no reconfiguration algorithm is used.
RTA-MAX+FA-VNR	It means that the FA-VNR algorithm is periodically used in the VNE process.
RTA-MAX+TA-VNR	It means that the TA-VNR algorithm, which is the topology-awareness VN reconfiguration algorithm proposed in [18], is periodically used in the VNE process.

<https://doi.org/10.1371/journal.pone.0207705.t002>

## Results analysis

**Performance comparison.** To compare the performance of the three algorithms, we set the values of parameters  $M_{cpu}$  and  $M_{bw}$  to 50, and give the comparison results in Figs 2 and 3.

From Fig 2, we can see that the proposed reconfiguration algorithm improves the acceptance ratio and is superior to the current VN reconfiguration algorithm. When the simulation time is 30000 time units, the acceptance ratios of RTA-MAX+FA-VNR, RTA-MAX+TA-VNR and RTA-MAX algorithms are 64.56%, 62.03%, and 61.14%, respectively. The former increased by 4.08% and 5.59% respectively compared with the latter two. The relationship between the acceptance ratio and time is mainly related to the arrival number and departure number of the VN requests. At the initial stage, the number of arrived VN requests is less and the available resources of the physical network are more abundant, so the acceptance ratio is higher. As the simulation time continues, the number of arrived VN requests gradually increases, and the available resources of the physical network gradually decrease, so the acceptance ratio rapidly declines. When the simulation time is to a certain extent, the arrival rate and the departure rate of the VN requests tend to be stable, and the available resources of the physical network tend to be stable, so the acceptance ratio tends to be stable. The reasons why the RTA-MAX+FA-VNR algorithm has the highest acceptance ratio are as follows. First, the FA-VNR algorithm dynamically adjusts the fragment degrees of the physical nodes through the migration mechanism to improve the load balancing of the physical network. Secondly,



**Fig 2. Acceptance ratio comparison.**

<https://doi.org/10.1371/journal.pone.0207705.g002>

the FA-VNR algorithm reduces the embedding cost of the embedded VNs through reconfiguration and releases some physical link resources.

As can be seen from Fig 3, the reconfiguration algorithms optimize the revenue to cost ratio of the physical network, and the effect of the proposed algorithm is more obvious. When the simulation time is 30000 time units, the revenue to cost ratios of RTA-MAX+FA-VNR, RTA-MAX+TA-VNR and RTA-MAX algorithms are 42.88%, 41.02%, and 39.60%, respectively. The former increased by 4.53% and 8.28% respectively compared with the latter two. Compared with other algorithms, the RTA-MAX+FA-VNR algorithm not only improves the average embedding revenue of the physical network, but also reduces the average embedding cost of the physical network, so it has the highest revenue to cost ratio. In the initial stage of simulation, the available resources of the physical network are abundant, resulting in better embedding schemes for the VN requests, so the revenue to cost ratios of all the three algorithms are higher. After a period of simulation, the arrival rate and the departure rate of the VN requests tend to be stable, and the available resources of the physical network also tend to be stable, resulting in the stability of the revenue to cost ratios of all the three algorithms.

**Evaluation with different resource requirement of VN requests.** In order to further verify the performance of the proposed reconfiguration algorithm, we investigate the influence of the bandwidth resource requirement and CPU resource requirement of the VN requests on the performance of it. When the value of  $M_{bw}$  increases from 10 to 90 while keeping the value of  $M_{cpu}$  at 50, the effects of the FA-VNR algorithm on the performance of the RTA-MAX algorithm in the stable state are shown in Figs 4 and 5.

From Fig 4, we can see that the proposed reconfiguration algorithm FA-VNR improves the acceptance ratio of the basic algorithm RTA-MAX. When parameter  $M_{bw}$  is less than 30, the optimization effect of the FA-VNR algorithm is not obvious. This is because when the maximum bandwidth requirement of the VN requests is small, the embedding failure of the VN requests is mainly caused by the failure of node embedding. When  $M_{bw}$  is greater than 40 and less than 70, the FA-VNR algorithm has a significant improvement in the acceptance ratio of the RTA-MAX algorithm, and the acceptance ratio of RTA-MAX+FA-VNR (0.6372 on average) is about 5.29% higher than that of RTA-MAX (0.6052 on average). When the parameter

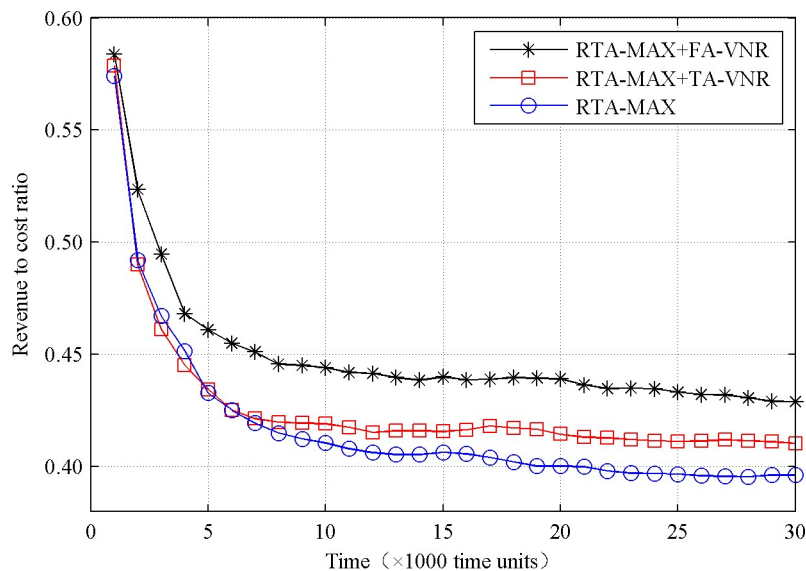


Fig 3. Revenue to cost ratio comparison.

<https://doi.org/10.1371/journal.pone.0207705.g003>

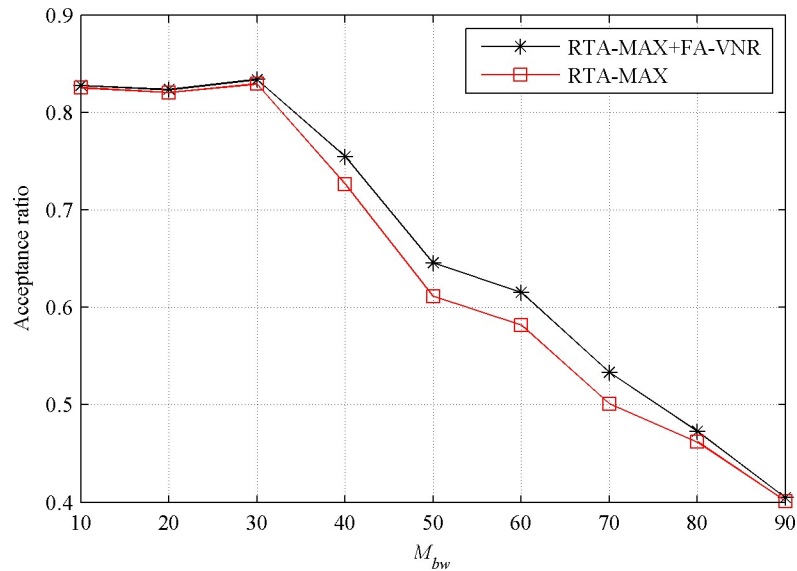


Fig 4. Acceptance ratio with increasing  $M_{bw}$ .

<https://doi.org/10.1371/journal.pone.0207705.g004>

$M_{bw}$  is greater than 80, the maximum bandwidth requirement of the virtual links is close to the maximum available bandwidth resource of the physical links. By reconfiguration, the physical link load cannot be changed well, so the FA-VNR algorithm has only a slight improvement in the acceptance ratio of the RTA-MAX algorithm.

From Fig 5, we can see that the proposed reconfiguration algorithm FA-VNR improves the revenue to cost ratio of the basic algorithm RTA-MAX. With the increase of  $M_{bw}$ , the improvement effect of the FA-VNR algorithm on the revenue to cost ratio of the RTA-MAX algorithm increases first and then decreases. When  $M_{bw}$  is greater than 40 and less than 70, the FA-VNR algorithm has a significant improvement in the revenue to cost ratio of the RTA-MAX

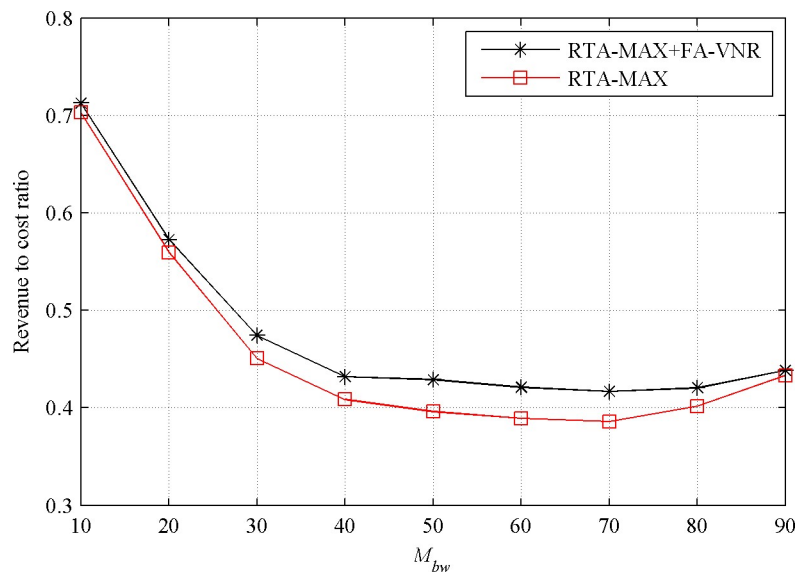
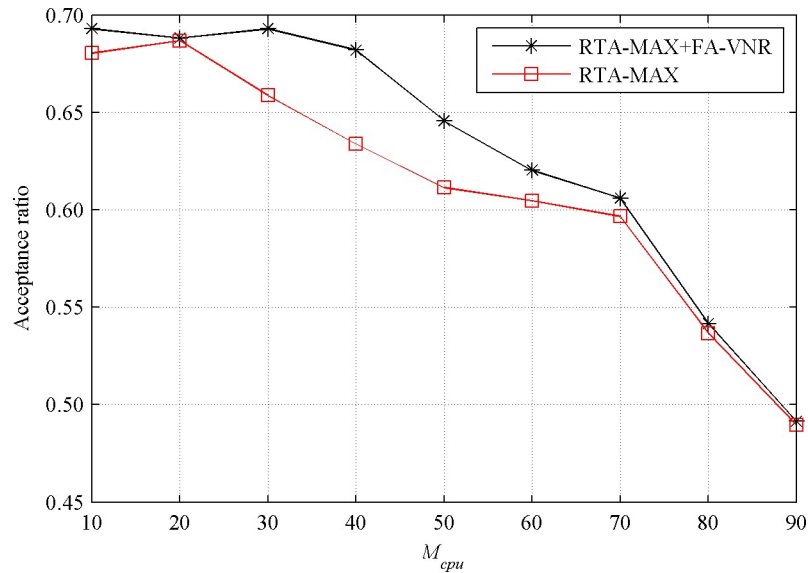


Fig 5. Revenue to cost ratio with increasing  $M_{bw}$ .

<https://doi.org/10.1371/journal.pone.0207705.g005>



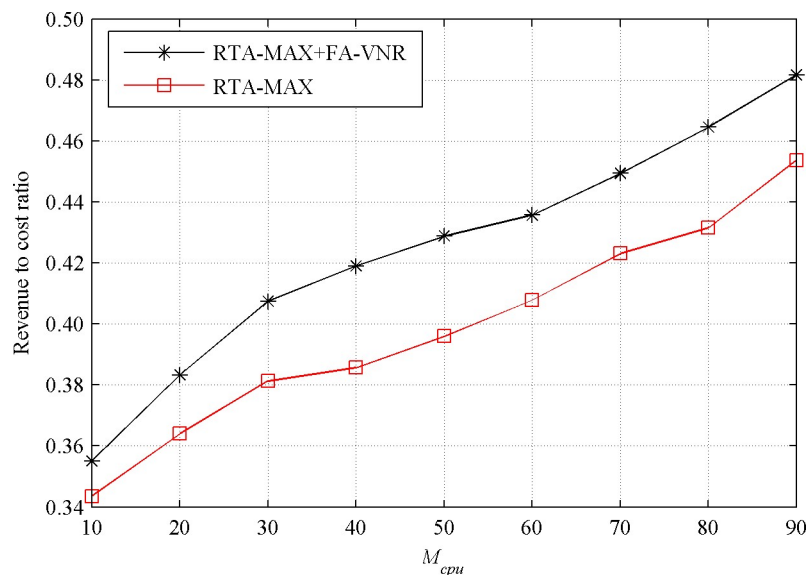
**Fig 6. Acceptance ratio with increasing  $M_{cpu}$ .**

<https://doi.org/10.1371/journal.pone.0207705.g006>

algorithm, and the revenue to cost ratio of RTA-MAX+FA-VNR (0.4246 on average) is about 7.52% higher than that of RTA-MAX (0.3949 on average).

When the value of  $M_{cpu}$  increases from 10 to 90 while keeping the value of  $M_{bw}$  at 50, the acceptance ratios and revenue to cost ratios of the RTA-MAX+FA-VNR and RTA-MAX algorithms in the stable state are shown in Figs 6 and 7, respectively.

From Fig 6, we can see that when the value of  $M_{cpu}$  is smaller (less than 20) or larger (greater than 60), the improvement effect of the FA-VNR algorithm on the acceptance ratio of the RTA-MAX algorithm is not obvious. When the  $M_{cpu}$  value is more than 30 and less than 50, the improvement effect is more obvious, and the acceptance ratio of RTA-MAX+FA-VNR (0.6735 on average) is about 6.12% higher than that of RTA-MAX (0.6346 on average).



**Fig 7. Revenue to cost ratio with increasing  $M_{cpu}$ .**

<https://doi.org/10.1371/journal.pone.0207705.g007>

From Fig 7, we can see that under all simulation conditions, the FA-VNR algorithm can significantly improve the revenue to cost ratio of the RTA-MAX algorithm, and the revenue to cost ratio of RTA-MAX+FA-VNR (0.4250 on average) is about 6.64% higher than that of RTA-MAX (0.3985 on average).

## Conclusion

In this paper, in order to improve the performance of the online VNE algorithm, a method to measure the fragment degree of the physical node is defined first, then an integer linear programming model for VN reconfiguration is constructed and a heuristic VN reconfiguration embedding algorithm FA-VNR is proposed. By periodically executing the FA-VNR algorithm, the fragment degrees of the physical nodes can be reduced and the embedding cost of the embedded VNs can be reduced, thus the acceptance ratio and the revenue to cost ratio of the basic algorithm can be improved. Through extensive simulation experiments, it is shown that when the maximum resources requirement of the VN requests are too large or too small, the FA-VNR algorithm has no obvious effect on the improvement of the basic algorithm RTA-MAX; when the maximum resources requirement of the VN requests are moderate, the FA-VNR algorithm can obviously improve the performance of the basic algorithm RTA-MAX. However, the latency of the physical and virtual networks has not been considered in this paper. To extend our work, we intend to introduce metrics such as latency, path length, and stress level into VN reconfiguration to provide QoS guaranteed virtual networks.

## Supporting information

**S1 Fig. Physical network topology.** This figure depicts the topology of the physical network in the simulations.

(TIF)

**S1 File. Initial data.** This document contains the initial data in the simulations.

(ZIP)

## Acknowledgments

We would like to thank the anonymous reviewers for their very helpful comments and feedbacks to improve the manuscript.

## Author Contributions

**Formal analysis:** Xinbo Liu, Buhong Wang.

**Methodology:** Xinbo Liu.

**Project administration:** Buhong Wang.

**Software:** Xinbo Liu.

**Writing – original draft:** Xinbo Liu.

**Writing – review & editing:** Buhong Wang.

## References

1. Anderson T, Peterson L, Shenker S, Turner J. Overcoming the Internet impasse through virtualization. *Computer*, 2005, 38(4): 34–41. <https://doi.org/10.1109/MC.2005.136>
2. Alcober J, Hesselbach X, de la Oliva A, Garcia-Saavedra A, Roldan D, Bock C. Internet future architectures for network and media independent services and protocols. In proceedings of the 15th

- International Conference on Transparent Optical Networks, Cartagena, Spain, 23–27 June 2013; pp.1-4. <https://doi.org/10.1109/ICTON.2013.6602680>
3. Razzaq A, Rathore MS. An approach towards resource efficient virtual network embedding. In proceedings of the 2nd International Conference on Evolving Internet, Valencia, Spain, 20–25 Sept. 2010; pp.68-73. 2010. <https://doi.org/10.1109/INTERNET.2010.21>
  4. Zhang S, Qian Z, Wu J, LU S Epstein L. Virtual network embedding with opportunistic resource sharing. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(3): 816–827. <https://doi.org/10.1109/TPDS.2013.64>
  5. Zhang P, Yao H, Liu Y. Virtual network embedding based on the degree and clustering coefficient information. IEEE Access, 2016, 4: 8572–8580. <https://doi.org/10.1109/ACCESS.2016.2632421>
  6. Ding J, Huang T, Liu J, Liu Y. Virtual network embedding based on real-time topological attributes. Frontiers of Information Technology & Electronic Engineering, 2015, 16(2):109–118.
  7. Yu C, Lian Q, Zhang D, Wu C. PAME: Evolutionary membrane computing for virtual network embedding. Journal of Parallel and Distributed Computing, 2018, 111: 136–151. <https://doi.org/10.1016/j.jpdc.2017.08.005>
  8. Rubio-Loyola J, Aguilar-Fuster C, Toscano-Pulido G, Mijumbi R, Serrat-Fernández J. Enhancing meta-heuristic-based online embedding in network virtualization environments. IEEE Transactions on Network and Service Management, 2018, 15(1): 200–216. <https://doi.org/10.1109/TNSM.2017.2742666>
  9. Zhang Z, Su S, Zhang J, Shuang K, Xu P. Energy aware virtual network embedding with dynamic demands: Online and offline. Computer Networks, 2015, 93: 448–459. <https://doi.org/10.1016/j.comnet.2015.09.036>
  10. FISCHER A, BOTERO JF, BECK MT, de Meer H, Hesselbach X. Virtual Network Embedding: A survey. IEEE Communications Surveys & Tutorials, 2013, 15(4): 1888–1906. <https://doi.org/10.1109/SURV.2013.013013.00155>
  11. Tran PN, Timm-Giel A. Reconfiguration of virtual network mapping considering service disruption. In proceedings of the IEEE International Conference on Communications, Budapest, Hungary, 9–13 June 2013; pp.3487-3492. <https://doi.org/10.1109/ICC.2013.6655090>
  12. Tran PN, Casucci L, Timm-Giel A. Optimal mapping of virtual networks considering reactive reconfiguration. In proceedings of the IEEE 1st International Conference on Cloud Networking, Paris, France, 28–30 Nov. 2012; pp.35-40. <https://doi.org/10.1109/CloudNet.2012.6483651>
  13. Hsu WH, Shieh YP, Wang CH, Yeh SC. Virtual network mapping through path splitting and migration. In proceedings of the 26th International Conference on Advanced Information Networking and Applications Workshops, Fukuoka, Japan, 26–29 March 2012; pp.1095-1100. <https://doi.org/10.1109/WAINA.2012.74>
  14. Mahboobeh Z, Aguilar-Fuster C, Rubio-Loyola J. A virtual network migration approach and analysis for enhanced online virtual network embedding. In proceedings of the 12th International Conference on Network and Service Management, Montreal, QC, Canada, 31 Oct.-4 Nov. 2016; pp.324-329. <https://doi.org/10.1109/CNSM.2016.7818441>
  15. Chowdhury SR, Ahmed R, Shahriar N, Khan A, Boutaba R, Mitra J, et al. ReViNE: Reallocation of Virtual Network Embedding to eliminate substrate bottlenecks. In proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management, Lisbon, Portugal, 8–12 May 2017; pp. 116–124. <https://doi.org/10.23919/INM.2017.7987271>
  16. Zhu H. A virtual network reconfiguration algorithm to improve acceptance ratio. 10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA), Krakow, Poland, 4–6 Nov. 2015; pp: 602–604. <https://doi.org/10.1109/BWCCA.2015.78>
  17. Qu H, Zhao J, Guo S, Wang H. Resource reconfiguration method based on minimum cost for network virtualization. Journal of Beijing University of Posts and Telecommunications, 2014, 37(5): 114–118.
  18. Peng L. A topology-awareness virtual network reconfiguration algorithm. Journal of Sichuan University (Engineering Science Edition), 2015, 47(5):110–115.
  19. Gong S, Chen J, Zhao S, Zhu Q. Virtual network embedding with multi-attribute node ranking based on TOPSIS. KSII Transactions on Internet and Information Systems, 2016, 10(2): 522–541.
  20. Liao S, Wu C, Zhang M, Jiang M. An efficient virtual network embedding algorithm with delay constraints. 16th International Symposium on Wireless Personal Multimedia Communications (WPMC), Atlantic City, NJ, USA, 24–27 June 2013.
  21. Yang Z, Guo Y. An exact virtual network embedding algorithm based on integer linear programming for virtual network request with location constraint. China Communications, 2016, 13(8):177–183. <https://doi.org/10.1109/CC.2016.7563720>
  22. Cao H, Yang L, Liu Z, Wu M. Exact solutions of VNE: A survey. China Communications, 2016, 13(6):48–62. <https://doi.org/10.1109/CC.2016.7513202>

23. Beck M T, Linnhoff-Popien C, Fischer A, Kokot F, de Meer H. A simulation framework for virtual network embedding algorithms. 16th International Telecommunications Network Strategy and Planning Symposium (Networks), Funchal, Portugal, 17–19 Sept. 2014. <https://doi.org/10.1109/NETWKS.2014.6959238>
24. <https://www.cc.gatech.edu/projects/gtitm/>
25. Zegura E W, Calvert K L, Bhattacharjee S. How to model an internetwork. Proceedings of IEEE INFOCOM '96. Conference on Computer Communications, San Francisco, CA, USA, 24–28 March 1996; pp: 594–602. <https://doi.org/10.1109/INFCOM.1996.493353>