Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

A forwarding graph embedding algorithm exploiting regional topology information[*]

Hong-chao HU[†‡], Fan ZHANG, Yu-xing MAO, Zhen-peng WANG
(*National Digital Switching System Engineering & Technological R&D Center, Zhengzhou 450002, China*)
[†]E-mail: huhongchao@gmail.com

Received July 8, 2016; Revision accepted Nov. 10, 2016; Crosschecked Nov. 24, 2017

Abstract: Network function virtualization (NFV) is a newly proposed technique designed to construct and manage network functions dynamically and efficiently. Allocating physical resources to the virtual network function forwarding graph is a critical issue in NFV. We formulate the forwarding graph embedding (FGE) problem as a binary integer programming problem, which aims to increase the revenue and decrease the cost to a service provider (SP) while considering limited network resources and the requirements of virtual functions. We then design a novel regional resource clustering metric to quantify the embedding potential of each substrate node and propose a topology-aware FGE algorithm called 'regional resource clustering FGE' (RRC-FGE). After implementing our algorithms in C++, simulation results showed that the total revenue was increased by more than 50 units and the acceptance ratio by more than 15%, and the cost of the service provider was decreased by more than 60 units.

Key words: Network function virtualization; Virtual network function; Forwarding graph embedding
https://doi.org/10.1631/FITEE.1601404 **CLC number:** TP393

1 Introduction

Most current network services are defined by statically combining a large number of network functions (NFs) such as network address translators (NATs), load balancers, firewalls, and intrusion detection systems (IDSs). Traditionally, these functions are provided by dedicated hardware platforms, which are expensive and difficult to maintain and upgrade (Mehraghdam *et al.*, 2014).

Network function virtualization (NFV) (ETSI NFVISG, 2013) is a rising idea that offers more flexibility in network service delivery by implementing NFs as virtual network function (VNF). Exploiting virtualization technologies, service providers (SPs) respond to network service requests from customers by developing network services at an abstract level using a VNF forwarding graph (FG) and then deploy the developed FGs within their datacenters. Fig. 1 shows two VNF FGs which define the sequence of VNFs that packets traverse and the logical connections between VNFs to fulfill corresponding network services. Network services are provided on demand as a service and are composed of one or multiple VNFs running on the top of virtual machines that are located on standard servers (Xia *et al.*, 2014). Once a traffic flow arrives at an SP's datacenter, it is steered through a number of VNFs in a specific order based on the SP's provisioning policy and service level agreement (SLA) (Koh *et al.*, 2007).

How to map a VNF FG onto a substrate network in an SP's datacenter efficiently is called the forwarding graph embedding (FGE) problem. FGE is a critical issue that affects both the performance of the network service and the resource utilization of the substrate network. Specifically, each VNF is mapped to one of the substrate nodes (servers), and the logical connection between two VNFs is mapped to a substrate path that connects the corresponding end nodes.

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 61309020 and 61521003)

ORCID: Hong-chao HU, http://orcid.org/0000-0002-9770-6610

Network service requests arrive at different time instances and demand to set up VNF FGs with different VNFs and lifetimes.

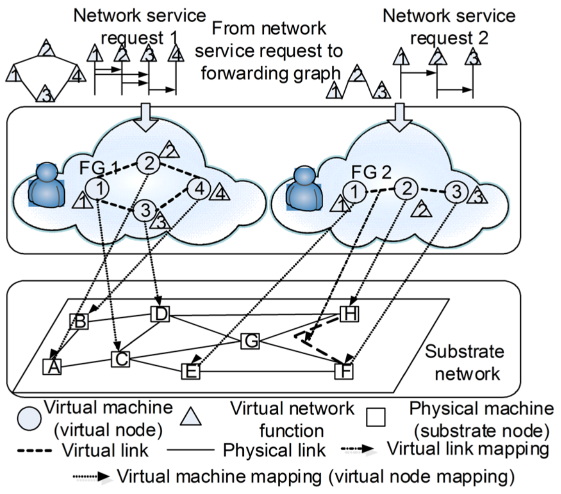
Network service request 1
From network service request to forwarding graph
Network service request 2
FG 1
FG 2
Substrate network
Virtual machine (virtual node)
Virtual network function
Physical machine (substrate node)
Virtual link
Physical link
Virtual link mapping
Virtual machine mapping (virtual node mapping)

Fig. 1 Mapping of forwarding graphs to a substrate network for network service requests

The FGE problem is similar to the virtual network embedding (VNE) problem, which aims to map virtual nodes and virtual links onto a substrate network. As a result, we know that the FGE problem is also NP-hard (Mehraghdam *et al.*, 2014). In the rest of this paper, virtual machines on which VNFs are running are referred to as virtual nodes, and the logical connections between them as virtual links. However, directly applying VNE solutions to the FGE problem is not practical. Firstly, current virtualization techniques do not have enough performance isolation between virtual machines on which VNFs are mapped. Hence, the behavior of one VNF can adversely affect the performance of another due to contention for shared physical resources (Pu *et al.*, 2010; Mei *et al.*, 2013). Bellavista *et al.* (2015) found that in industry-relevant cloud platforms, there may be computing overhead due to the intense communication load of virtual machines (VMs) co-located on the same physical host, and they proposed a VM placement algorithm to take this factor into consideration. Shea *et al.* (2014) found that there is degradation and variation in VM communications in lightly used cloud networks. They showed that such variation and degradation are due mainly to the dual-role of the CPU in both computation and network communication among VMs in the server. As a result, we conclude that VNFs with performance interference should be mapped to different substrate nodes. This is not the case in virtual network resource allocation areas. Secondly, virtual network (VN) embedding algorithms usually avoid mapping multiple virtual nodes from the same VN request onto one substrate node, which is different from the FGE problem. As a result, the solution space in FGE problems is larger than in VNE problems. Previous VNE algorithms that also have some shortcomings in balancing performance and computational complexity; hence, an efficient FGE algorithm is needed.

This study presents an efficient topology-aware solution to the FGE problem and aims to increase an SP's revenue while satisfying network service requirements. The main contributions of this paper are as follows: (1) We formulate the FGE problem as a binary integer programming (BIP) problem, which takes resource capacity of the substrate network and resource constraints of FGs into consideration and aims to increase long-term revenue and decrease the cost of the SP simultaneously. (2) We design a regional resource clustering metric to evaluate the embedding potential of substrate nodes and propose an efficient topology-aware heuristic FG embedding algorithm based on that metric. The algorithm takes both regional topology information and the energy consumption of the substrate network into consideration. (3) We propose a genetic algorithm based heuristic algorithm to maximize the long-term economic benefits to the SP.

2 Related work

NFV is a newly proposed concept that is still under investigation. Standardization of the problem definition and applications is not yet completed. To the best of our knowledge, very few studies have focused on our area of research. Xia *et al.* (2014) formulated the VNF placement problem as a binary integer programming problem. They regarded CPU resource as a constraint and therefore aimed to minimize the number of substrate nodes used to host the required VNFs. However, they did not consider the bandwidth constraints of the substrate links, which are also an important factor. Mehraghdam *et al.* (2014) described the VNF placement problem as a mixed integer quadratically constrained program. They considered the resource capacity and resource requirements of both the substrate network and VNFs. However, they did not propose any heuristic algorithm, since solving mixed-integer programming (MIP) problems is known to be computationally intractable. Thus, using the above methods to solve the VNF placement problem is impracticable. Bellavista *et al.* (2015) found that in industry-relevant cloud platforms, there may be a computing overhead due to the intense communication load of VMs co-located on the same physical host. They proposed a VM placement algorithm that not only satisfies the predicted communication demands, but also takes into account the computing overhead due to resource contention of the co-located VMs.

NFV provides an on-demand and per-flow network service for customers' network traffic, which significantly improves compatibility and flexibility, and introduces the FGE problem. Some previous studies have focused on the virtual network embedding problem, which is similar to the FGE problem in some respects. In simple terms, in those studies the problem space was restricted by assuming that virtual network requests are known in advance (Fan and Ammar, 2006; Lu and Turner, 2006) and focused on the problem either in the absence of node or link constraints or presuming infinite resource capacity of the substrate networks (Fan and Ammar, 2006; Lu and Turner, 2006; Razzaq and Siraj Rathore, 2010).

Dealing with the full problem space, in some studies node mapping and link mapping were solved separately to reduce the solution space (Lu and Turner, 2006; Yu *et al.*, 2008; Lischka and Karl, 2009; Su *et al.*, 2012). However, two-stage algorithms can result in poor performance because of their ignorance of the relationship between node mapping and link mapping. To overcome the shortcomings introduced by two-stage embedding algorithms, Chowdhury *et al.* (2009) and Cheng *et al.* (2011) conducted node mapping and link mapping at the same stage, thereby producing one-stage embedding algorithms. Although they achieve better performance, these algorithms also increase computational complexity.

In recent studies (Bhatia *et al.*, 2008; Zhang *et al.*, 2012; Gong *et al.*, 2014) topology-aware ranking metrics have been presented to identify the bottleneck nodes and links in substrate networks, and a set of algorithms have been proposed based on those metrics. Although solving node mapping and link mapping separately, these algorithms consider link mapping constraints in the node mapping stage and achieve a better tradeoff between performance and computational complexity.

The existing VNE algorithms cannot be applied directly to the FGE problem for the following reasons: (1) Current virtualization techniques do not have enough performance isolation between virtual machines, so different VNFs may have performance interference. Hence, VNFs contending for the same type of physical resources cannot be co-located on one substrate node, which is not considered in VNE algorithms. (2) In VNE problems, virtual nodes from the same virtual network request cannot be mapped to one substrate node. However, the FGE problem does not have such a constraint and has a larger solution space. As a result, the FGE problem needs more efficient mapping algorithms.

3 Key concepts and preliminaries

The FGE problem is similar to the virtual network embedding problem in the following three respects: (1) In the FGE problem, the requested VNFs should be mapped onto physical/substrate nodes, and logical connections between VNFs should be mapped to physical/substrate paths, where nodes/links resource requirements should be satisfied. In the virtual network embedding problem, virtual nodes and virtual links should also be mapped onto physical servers and links, ensuring certain constraints are met. (2) The objectives of the FGE problem and virtual network embedding problem are to increase the long-term revenue of SPs, and reduce the cost of embedding each forwarding graph and virtual network. (3) Both of these problems are NP-hard.

3.1 Substrate network

We consider a substrate network consisting of multiple commodity servers connected by a network. Each commodity server is able to perform computing tasks and forward packets. In the rest of this paper, we use substrate node (SN) and substrate link (SL) to represent the commodity server and physical links between two servers, respectively.

Definition 1 (Substrate network) Similar to previous work (Chowdhury *et al.*, 2009; Lischka and Karl, 2009), an SN is modeled as a weighted undirected graph denoted by $G_s=(V_s, E_s)$, where V_s and E_s refer to the sets of substrate nodes and links, respectively. Each substrate node $v_s \in V_s$ is associated with a CPU capacity value $c_s(v_s)$. Each substrate link e_s is associated with a bandwidth capacity value $b_s(e_s)$ and a link delay value $d_s(e_s)$. Fig. 2c shows an example of an SN. The numbers over the nodes and links indicate the available CPU and bandwidth resources, respectively.

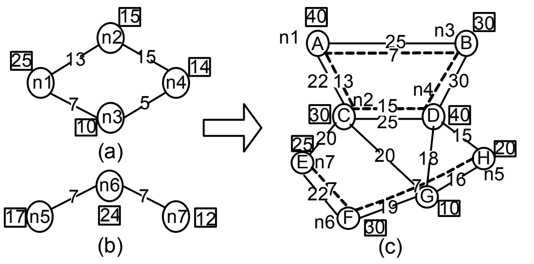

Fig. 2 Resource allocation of forwarding graphs: (a) forwarding graph 1; (b) forwarding graph 2; (c) substrate network

3.2 Virtual network function forwarding

A VNF FG defines the sequence of VNFs packeting traverse and the logical connection between VNFs. Each VNF runs on one VM. In the rest of this paper, we refer to a VM as a VN and the connections between VMs as virtual links (VLs).

Definition 2 (Virtual network function forwarding graph) A VNF FG $F_r=(G_r, T_a, T_d)$ is a three tuple, similar to that described by Yu *et al.* (2008). $G_r=(V_r, E_r)$ is a weighted undirected graph indicating the topology of the FG, where V_r and E_r refer to the set of virtual nodes (VNs) and virtual links (VLs), respectively. Suppose there are n types of VNFs and each virtual node $v_r \in V_r$ has a VNF type in those n types. Each VNF type i has an interference vector $\boldsymbol{r}_i=(r(1), r(2), \ldots, r(n))$, where $r_i(j) \in \{0, 1\}$ has value 1 if the VNF is with type i and the VNF with type j can be collocated on the same SN; otherwise, its value is set to 0. A virtual node is also associated with a CPU constraint $c_r(v_r)$. Each virtual link $e_r \in E_r$ is associated with a bandwidth constraint $b_r(e_r)$ and a link delay constraint $d_r(e_r)$. T_a indicates the arriving time and T_d the duration of the FG. Figs. 2a and 2b present two FGs with node and link requirements.

The notations used in this paper (Table 1) is similar to those used by Gong *et al.* (2014). The subscript indicates the substrate network/forwarding graph.

Table 1 Notations of the forwarding graph model

Notation	Meaning
G_s	Topology of the substrate network
V_s	Set of substrate nodes
E_s	Set of substrate links
$c_s(v_s)$	CPU capacity of substrate node v_s
$b_s(v_s)$	Bandwidth capacity of substrate link e_s
$d_s(e_s)$	Link delay of substrate link e_s
RC_s	Set of residual CPU resources
RB_s	Set of residual bandwidth resources
P_s	Set of loop-free substrate paths
F_r	Virtual network function forwarding graph
G_r	Topology of the forwarding graph
V_r	Set of virtual nodes
E_r	Set of virtual links
$c_r(v_r)$	CPU constraints of virtual node v_r
$b_r(e_r)$	Bandwidth of virtual link e_r
$d_r(e_r)$	Delay constraint of virtual link e_r
$\boldsymbol{r}_i$	Interference vector of VNF type i

3.3 Forwarding graph embedding

When an FG is developed, the SP has to decide whether to accept it. If the FG is accepted, the SP then allocates a proper amount of resources to satisfy the FG's constraints. Once the FG expires, the resources assigned to that FG are released.

Definition 3 (Forwarding graph embedding) Similar to previous work (Chowdhury *et al.*, 2009; Lischka and Karl, 2009), a forwarding graph embedding of an FG to the substrate network is defined as a mapping M from G_r to a subset of G_s, M: $G_r \rightarrow G_s'$ (V_s', E_s'), where $V_s' \subseteq V_s$, $E_s' \subseteq E_s$. The embedding process can be decomposed into two major components: node embedding M_n and link embedding M_l.

Node embedding M_n maps a virtual node to a substrate node with enough available computing resources to meet its demands. Multiple virtual nodes of an FG can be co-located on the same substrate node as long as they do not have performance interference. Mathematically, the node mapping can be described as a many-to-one mapping, i.e.,

$$M_n : V_r \rightarrow V_s', \text{ subjected to:}$$

$$\forall n_{\mathrm{r}}, m_{\mathrm{r}} \in V_{\mathrm{r}},\ \text{if } r_{n_{\mathrm{r}},m_{\mathrm{r}}}=0$$
$$\begin{cases} M_{\mathrm{n}}(n_{\mathrm{r}}) \in V_{\mathrm{s}}, \\ M_{\mathrm{n}}(n_{\mathrm{r}}) \neq M_{\mathrm{n}}(m_{\mathrm{r}}), \\ \mathrm{RC}_s(M_{\mathrm{n}}(n_{\mathrm{r}})) \geq c_{\mathrm{r}}(n_{\mathrm{r}}). \end{cases}$$

The link embedding M_{l} maps a virtual link to a loop-free substrate path in a substrate network consisting of at least one substrate link. Similar to node embedding, the link embedding can be described as

$$M_{\mathrm{l}}: E_{\mathrm{r}} \to P_{\mathrm{s}},\ \text{subjected to:}$$
$$\forall e_{\mathrm{r}} = (m_{\mathrm{r}} n_{\mathrm{r}}) \in E_{\mathrm{r}}$$
$$\begin{cases} M_{\mathrm{l}}(m_{\mathrm{r}} n_{\mathrm{r}}) \in P_{\mathrm{s}}(M_{\mathrm{n}}(m_{\mathrm{r}}), M_{\mathrm{n}}(n_{\mathrm{r}})), \\ \mathrm{RB}_{\mathrm{s}}(M_{\mathrm{l}}(m_{\mathrm{r}} n_{\mathrm{r}})) \geq b_{\mathrm{r}}(e_{\mathrm{r}}), \end{cases}$$

where P_{s} is the set of loop-free paths between substrate nodes hosting the end nodes of the virtual link. Fig. 2c shows that FG1 and FG2 can be mapped onto the substrate network. For FG1, we have

$$\begin{aligned} M_{\mathrm{N}} &= \{n_1 \to A, n_2 \to C, n_3 \to B, n_4 \to D\}, \\ M_{\mathrm{L}} &= \{e(n_1,n_2) \to (A,C), e(n_2,n_4) \to (C,D), \\ &\quad e(n_1,n_3) \to (A,B), e(n_3,n_4) \to (B,D)\}. \end{aligned}$$

3.4 Objectives

As FG embedding is done by an SP, the main objective of FG embedding is to increase revenue and decrease the cost of embedding the forwarding graph. A customer is willing to pay only for the resources he/she requires. As a result, the revenue of embedding an FG at time t can be defined as

$$\mathrm{Rvn}(F_{\mathrm{r}},t) = T_{\mathrm{d}}\Big(w_{\mathrm{c}} \sum_{v_{\mathrm{r}} \in V_{\mathrm{r}}} c_{\mathrm{r}}(v_{\mathrm{r}}) + w_{\mathrm{b}} \sum_{e_{\mathrm{r}} \in E_{\mathrm{r}}} b_{\mathrm{r}}(e_{\mathrm{r}})\Big), \quad (1)$$

where w_{c} and w_{b} are the unit prices of CPU and bandwidth resources, respectively.

Since the SP maps FGs to the substrate network, there is also a mapping cost, consisting of a deployment cost and an operation cost at time t, which can be defined as follows:

$$\mathrm{Cost}(F_{\mathrm{r}},t) = \mathrm{Cost}_{\mathrm{d}}(F_{\mathrm{r}},t) + \mathrm{Cost}_{\mathrm{o}}(F_{\mathrm{r}},t), \quad (2)$$

where $\mathrm{Cost}_{\mathrm{d}}(F_{\mathrm{r}}, t)$ refers to the deployment cost and $\mathrm{Cost}_{\mathrm{o}}(F_{\mathrm{r}}, t)$ to the operation cost for mapping FG F_{r} at time t. Similar to Chowdhury *et al.* (2009), the deployment cost of successfully embedding an FG is defined as being proportional to the sum of the total resources assigned to that FG:

$$\mathrm{Cost}_{\mathrm{d}}(F_{\mathrm{r}},t) = T_{\mathrm{d}}\left(\sum_{v_{\mathrm{r}} \in V_{\mathrm{r}}} c_{\mathrm{r}}(v_{\mathrm{r}}) + \sum_{e_{\mathrm{r}} \in E_{\mathrm{r}}} b_{\mathrm{r}}(e_{\mathrm{r}})\,|M_{\mathrm{l}}(e_{\mathrm{r}})|\right), \quad (3)$$

where $|M_{\mathrm{l}}(e_{\mathrm{r}})|$ is the length of the substrate path which carries the virtual link e_{r}.

In this study, we consider the operation cost for mapping the FG F_{r} to be the energy consumed by the substrate nodes to host the virtual nodes of that FG, and the energy consumed to run the substrate server. For simplicity, we assume that the servers are homogeneous with regard to their energy consumption. The operation cost is defined as

$$\mathrm{Cost}_{\mathrm{o}}(F_{\mathrm{r}},t) = T_{\mathrm{d}} \cdot \sum_{v_{\mathrm{r}} \in V_{\mathrm{r}}} \left\{ \frac{p_{\mathrm{b}} \cdot c_{\mathrm{r}}(v_{\mathrm{r}})}{c_{\mathrm{s}}[M_{\mathrm{n}}(v_{\mathrm{r}})] - \mathrm{RC}_{\mathrm{s}}[M_{\mathrm{n}}(v_{\mathrm{r}})]} + p_{\mathrm{a}} \cdot c_{\mathrm{r}}(v_{\mathrm{r}}) \right\}, \quad (4)$$

where p_{b} is the baseline energy consumption in each time unit of each substrate node if it is in the ‘on’ state, and p_{a} is the energy consumption of each unit of CPU requirement. According to Eq. (4), if a substrate node has no virtual nodes on it, it will contribute no energy consumption to the operation cost.

From the above definitions, the revenue is fixed if an FG is given. Therefore, the SP should try to accept more FGs and allocate proper resources to them, decreasing the cost of embedding each FG and increasing profits.

As in Zhang *et al.* (2012), the long-term average revenue is defined by

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} \mathrm{Rvn}(F_{\mathrm{r}},t). \quad (5)$$

The FG acceptance ratio of the substrate network can be defined by

$$\lim_{T\to\infty} \sum_{t=0}^{T} \mathrm{FG}_{\mathrm{s}}(t) \Big/ \sum_{t=0}^{T} \mathrm{FG}(t), \quad (6)$$

where $\mathrm{FG_s}$ is the number of FGs successfully accepted by the substrate network at time t, and FG is the total number of arrival FGs at time t.

The long-term revenue to cost ratio is also considered in measuring the efficiency of resource utilization in the substrate network:

$$\lim_{T\to\infty}\sum\nolimits_{t=0}^{T}\mathrm{Rvn}(F_\mathrm{r},t)\Big/\sum\nolimits_{t=0}^{T}\mathrm{Cost}(F_\mathrm{r},t). \quad (7)$$

Higher FG acceptance and R/C ratios are preferred if the long-term average revenues of the FG embedding solution are about the same.

4 Binary integer programming formulation for forwarding graph embedding

In this section we formulate the forwarding graph problem as a binary integer programming problem (BIP), in which each virtual node must be mapped to exactly one substrate node. Each substrate node is able to host multiple virtual nodes from the same FG if these virtual nodes do not have performance interference. We consider a situation where virtual links are unsplittable and therefore cannot be embedded to multiple substrate paths. We present the following BIP formulation.

FGE_ILP

1. Variables

$f_{uv}^{ij}\in\{0,1\}$: A binary variable denoting whether the substrate path hosting virtual link (i, j) passes substrate link (u, v). If the substrate path passes the substrate link, the variable is set to 1 and 0 otherwise.

$x_u^i\in\{0,1\}$: A binary variable, which has the value 1 if virtual node i is mapped to substrate node u, and 0 otherwise.

2. Objectives

Minimize

$$\sum_{i\in V_\mathrm{r}}\sum_{u\in V_\mathrm{s}}\frac{p_\mathrm{b}\cdot c_\mathrm{r}(i)}{[c_\mathrm{s}(u)-\mathrm{RC_s}(u)]\cdot\mathrm{RC_s}(u)+\delta} + \sum_{l_{uv}\in E_\mathrm{s}}\frac{1}{\mathrm{RB_s}(l_{uv})+\delta}\sum_{l_{ij}\in E_\mathrm{r}}f_{uv}^{ij}. \quad (8)$$

3. Constraints

Capacity constraints:

$$\sum_{l_{ij}\in E_\mathrm{r}}f_{uv}^{ij}\cdot b_\mathrm{r}(l_{ij})\le\mathrm{RB_s}(l_{uv}),\ \forall u,v\in V_\mathrm{s},\forall i,j\in V_\mathrm{r}. \quad (9)$$

$$\sum_{i\in V_\mathrm{r}}x_u^i\cdot c_\mathrm{r}(i)\le\mathrm{RC_s}(u),\ \forall u\in V_\mathrm{s},\forall i\in V_\mathrm{r}. \quad (10)$$

Flow-related constraints:

$$\sum_{l_{uk}\in E_\mathrm{s}}f_{uk}^{ij}-\sum_{l_{kv}\in E_\mathrm{s}}f_{vk}^{ij}=x_u^i-x_v^i,\ \forall u\in V_\mathrm{s},\forall i,j\in V_\mathrm{r}. \quad (11)$$

Link delay constraints:

$$\sum_{l_{uv}\in E_\mathrm{s}}d_\mathrm{s}(l_{uv})\cdot f_{uv}^{ij}\le d_\mathrm{r}(l_{ij}),\ \forall u,v\in V_\mathrm{s},\forall i,j\in V_\mathrm{r}. \quad (12)$$

Constraints to ensure that a virtual node is mapped to exactly one substrate node:

$$\sum_{u\in V_\mathrm{s}}x_u^i=1,\ \forall i\in V_\mathrm{r},\forall u\in V_\mathrm{s}. \quad (13)$$

Performance interference constraints:

$$x_u^i\cdot x_u^j\le r_{ij},\ \forall u\in V_\mathrm{s},\forall i,j\in V_\mathrm{r}. \quad (14)$$

4. Explanations of the formulation

The objective function of the BIP in Eq. (8) tries not only to minimize the cost of embedding the FG but also to balance the load. In the first part of the formulation, the virtual node operation cost is divided by the product of the residual CPU capacity and allocated CPU capacity of the substrate node to which the virtual node is mapped. This ensures that substrate nodes with more residual CPU capacity and to which more CPU capacity has already been allocated are preferred. Similarly, the second part of the formulation ensures that substrate links with more residual bandwidth are preferred. We ignore the deployment cost of the virtual nodes in the formulation as it is fixed once the virtual nodes are given. δ is a small positive constant to avoid being divided by zero in computing the objective function.

Constraints in Eqs. (9) and (10) ensure that the bandwidth and CPU resources allocated from the substrate network remain within their limitations.

Constraint in Eq. (11) refers to the flow conservation conditions.

Constraint in Eq. (12) ensures that the total delay of the path to which the virtual link is mapped does not violate its constraint.

Constraint in Eq. (13) denotes that a virtual node is mapped to exactly one substrate node and constraint in Eq. (14) ensures that a virtual node does not affect the performance of other virtual nodes mapped to the same substrate node.

5 Topology-aware heuristic forwarding graph embedding algorithm based on a regional resource clustering metric

As solving BIP is NP-hard (Chen *et al.*, 2012), it is not practical to solve this problem when the problem is large. Hence, we propose a topology-aware heuristic FG embedding algorithm called RRC-FGE (regional resource clustering FGE), which aims to decrease deployment and operation costs by taking both the regional topology information and the degree of resource aggregation into consideration. First, to evaluate the embedding potential of substrate nodes, we design a regional resource clustering metric *R*, which quantifies the embedding potential of substrate nodes. Second, to decrease further the embedding cost, we propose a comprehensive measurement NM to make a balance between the regional resource clustering metric and the operation cost of substrate nodes. We then conduct a greedy node mapping based on NM and a shortest-path based link mapping.

5.1 Regional resource clustering metric

Motivations: Some earlier studies considered link mapping constraints in the node mapping stage by proposing metrics to measure topology information and the global resource capacity of the substrate nodes (Bhatia *et al.*, 2008; Zhang *et al.*, 2012; Gong *et al.*, 2014). However, these metrics have the following disadvantages:

1. The adjacency limitation problem. These proposed global resource metrics quantify the embedding potential value of the substrate node by its available resources together with embedding potential values transferred by its incident links from other nodes. For example, as shown in Fig. 3, the nodes filled in grey in Figs. 3a and 3b have the same value in the above global resource metrics. However, the grey-filled node on the right has more available resources around it than that on the left, as there are links between the adjacent nodes of the grey-filled node in the figure on the right.

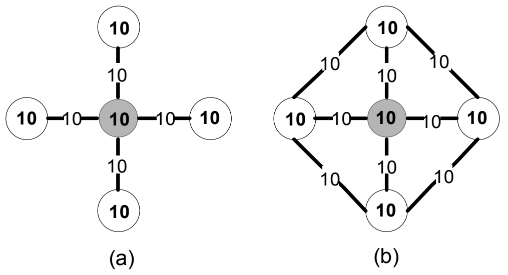

Fig. 3 An example of disadvantages of global resource metrics: (a) a network where the white nodes are not adjacent to each other; (b) a network where the white nodes are adjacent to each other

2. The regional pertinence absence problem. The proposed metrics in the above studies measured the embedding potential of a substrate node by considering resource information from all the other substrate nodes. In practice, virtual networks often have a much smaller size than substrate networks and should be compactly mapped to a small part of the substrate network. However, the metrics above lack the ability to measure the embedding potential of a node in relation to the topology information in its regional area.

Definition: To overcome the problems of the above mentioned metrics, we propose a regional resource clustering metric (RRC) to quantify the embedding potential of the nodes in substrate networks. This metric was inspired by semi-local centrality (Fagiolo, 2007) and the clustering coefficient (Eppstein, 1998) of a node. Specifically, given a network topology $G(V, E)$, the CPU capacity $c(u)$, $u \in V$, and the bandwidth resource $b(u, v)$, $(u, v) \in E$, the RRC value $R(u)$ of node u are formulated as

$$R(u) = \mathrm{Rf}(u) \cdot \ln[\mathrm{Cf}(u)], \tag{15}$$

where $\mathrm{Rf}(u)$ is a resource factor and $\ln[\mathrm{Cf}(u)]$ is a clustering factor of node u. We use $\ln[\mathrm{Cf}(u)]$ instead of $\mathrm{Cf}(u)$ for two reasons: (1) The value of the clustering factor is reduced. Thus, the resource factor contributes more to $R(u)$. (2) Using the logarithm can still guarantee a right result because a larger Cf(u) leads to a larger ln[Cf(u)].

$$\mathrm{Rf}(u)=d\cdot\overline{c(u)}+(1-d)\sum_{v\in\mathrm{Nei}(u)}\frac{b(u,v)}{\overline{b(v)}}\cdot Q(v),\quad(16)$$

$$Q(v)=d\cdot\overline{c(v)}+(1-d)\sum_{w\in\mathrm{Nei}(v)}\frac{b(v,w)}{\overline{b(w)}}\cdot\overline{c(w)},\quad(17)$$

$$\overline{c(u)}=\frac{c(u)}{\sum_{v\in V}c(v)},\quad(18)$$

$$\overline{b(u)}=\sum_{v\in\mathrm{Nei}}b(u,v),\quad(19)$$

where Nei(u) is the node set containing all the adjacent nodes of u, and $d\in(0, 1)$ is a constant number.

$$\mathrm{Cf}(u)=\frac{[W^{[1/3]}]^3_{|u||u|}}{d_u(d_u-1)},\quad(20)$$

$$W^{[1/3]}=\{\sqrt[3]{w_{ij}}\},\ \forall i,j\in V,\quad(21)$$

$$w_{ij}=\frac{b(i,j)}{\max(bw)},\ (u,v)\in E,\quad(22)$$

where $[W^{[1/3]}]^3_{|u||u|}$ is the uth element of the main diagonal of $[W^{[1/3]}]^3=W^{[1/3]}W^{[1/3]}W^{[1/3]}$, d_u is the degree of node u, $W^{[1/3]}$ is the matrix obtained from W by taking the 3rd root of each entry, and max(bw) is the largest bandwidth in the network.

Physical meaning: Eqs. (15)–(19) indicate that the resource factor of any node $u\in V$ is a weighted summation of the resource of its two-tier neighbors; i.e., other nodes in that region also contribute to the resource factor of node u. Eqs. (20)–(22) indicate that the clustering coefficient factor of any node $u\in V$ measures the bandwidth resources between its neighbors. The RRC metric combines the above two factors to overcome the drawbacks of previous metrics.

Calculation: The complexity of calculating the RRC is $O(|V|\cdot D^2(V))$, where $|V|$ and $D(V)$ are the number and average degree of the nodes in the network, respectively. We propose the algorithm for calculating the regional resource clustering metric as Algorithm 1.

Algorithm 1 Calculation of the regional resource clustering metric vector

```
1  $R\leftarrow 0$;
2  $\mathrm{Rf}_i=d\cdot\overline{c(i)}$;
3  for each $j\in$neighbour($i$) do
4      $q_j=d\cdot\overline{c(j)}$;
5      for each $k\in$neighbour($j$) do
6          if $k$==1
7              continue;
8          end if
9          $q_j+=(1-d)\frac{b(j,k)}{\overline{b(k)}}\overline{c(k)}$;
10     end for
11     $\mathrm{Rf}_i+=(1-d)\frac{b(i,j)}{\overline{b(j)}}q_j$;
12     $R_i=\mathrm{Rf}_i\cdot\ln\mathrm{Cf}(i)$;
13 end for
```

In Algorithm 1, the input is network topology $G(V, E)$ and output is an RRC vector $\boldsymbol{R}$.

However, as illustrated in Eq. (4), the operation cost for a substrate node to perform a unit computing task is inversely proportional to its allocated resources. Hence, to decrease the operation cost we should map virtual nodes to substrate nodes which have more allocated resources.

Let $\sigma\in(0, 1)$ be a constant number. To trade off between these two potentially conflicting goals, we measure the substrate nodes comprehensively as

$$\mathrm{NM}(u)=\sigma\cdot R(u)+(1-\sigma)[c(u)-\mathrm{RC}(r)].\quad(23)$$

5.2 Greedy node mapping

We adopt a greedy mapping method in the node mapping stage. After calculating the NM value of the substrate node, we sort virtual nodes in ascending order according to the number of virtual nodes with which each has performance interference. Then, we sort the virtual nodes that have the same interference number in descending order according to their CPU demands. For each virtual node, the algorithm selects a set of substrate nodes with enough available CPU resources to meet the virtual node's CPU demand and does not host any virtual node that has performance interference with it. The algorithm maps the virtual node to the substrate node with the highest NM value in that set. If there is no qualified substrate node in the set, the FG is blocked.

Algorithm 2 shows the details of the greedy node mapping algorithm. The time complexity of the algorithm is $O(|V_\mathrm{s}||V_\mathrm{r}|)$. The inputs are FG topology $G_\mathrm{r}(V_\mathrm{r}, E_\mathrm{r})$, substrate network $G_\mathrm{s}(V_\mathrm{s}, E_\mathrm{s})$, and the RRC vector $\boldsymbol{r}_\mathrm{s}$ for substrate network. The output is the node mapping M_n.

Algorithm 2 Greedy node mapping

```
1  Sort virtual nodes according to their interference
     number and CPU demand to obtain $V_\mathrm{r}^\mathrm{sort}$;
2  for each $v \in V_\mathrm{r}^\mathrm{sort}$ do
3      $\Phi_v \leftarrow \varnothing$;
4      for each $u \in V_\mathrm{s}$ do
5          if qualified_host($u$, $v$)=true
6              $\Phi_v \leftarrow \Phi_v \cup u$;
7          end if
8      end for
9      if $\Phi_v = \varnothing$
10         block $G_\mathrm{r}$;
11     else
12         $M_N(v) = \arg\max_{z \in \Phi_v}\{r_\mathrm{s}(z)\}$;
13     end if
14 end for
```

Function qualified_host(u, v) checks whether substrate node u is qualified to the host virtual node v. If $\mathrm{RC}_\mathrm{s} \geq c_\mathrm{s}(v)$ and u does not host any virtual node that has performance interference with v, the function returns true; otherwise, it returns false.

5.3 Link mapping

In the link mapping stage, our algorithm maps virtual links to substrate paths. For each virtual link, the algorithm first cuts off all the substrate links that violate its bandwidth constraint. Since virtual links are unsplittable, the algorithm finds the shortest path by searching the k-shortest paths (Wang *et al.*, 2014) on values of k, until a path is found to satisfy the delay requirement of the virtual link.

6 Genetic algorithm based forwarding graph embedding algorithm

In this section, we propose a genetic algorithm based heuristic (Goldberg, 2009), FGR-GA, to solve the NP-hard forwarding graph embedding problem. FGR-GA works as follows. Once the forwarding graph request arrives, forwarding graph regional-genetic algorithm (FGR-GA) randomly generates an initial population of feasible virtual node embedding solutions (i.e., chromosomes). During each iteration, the population evolves through a set of chromosome operators: selection, crossover, and mutation. After a given number of iterations, an optimal solution is selected to satisfy the request's demands.

6.1 Chromosome coding and initial population

In this study, we encode a virtual node embedding solution as a chromosome. The number of genes in the chromosome equals that of virtual nodes in the FG, and the value of each gene denotes the substrate node to which this virtual node is mapped. A chromosome is expressed as follows:

$$C = [g_1, g_2, ..., g_l], \tag{24}$$

where $l=|V_\mathrm{r}|$, and $g_k \in \{0, 1, ..., |V_\mathrm{s}|-1\}$ (k=1, 2, …, l).

FGR-GA generates randomly a large initial population of chromosomes. However, some of these chromosomes may not be feasible due to performance interference and substrate node capacity limitations. We propose a feasibility check method to make sure that all the chromosomes are feasible solutions to the node mapping problem. If a chromosome fails to pass the feasibility check, the gene that violates the constraints will select randomly another substrate node until the constraints are satisfied.

6.2 Crossover and mutation

In this study, we adopt the multi-point crossover method. In each iteration, the algorithm selects several chromosomes to be the parent chromosomes according to the crossover probability P_c. The genes denoted by the crossover point in each pair of parent chromosomes are swapped to generate child chromosomes.

The mutation operation is applied to avoid a local optimum; the goal is obtained by introducing a small percentage of randomness according to the mutation probability P_m.

After crossover and mutation operations, a feasibility check method similar to the feasibility check mentioned above is needed.

6.3 Fitness evaluation and chromosome selection

The fitness of a chromosome is an indicator of the quality of the virtual node mapping solution. We use the objective function in Eq. (8) as a fitness metric.

For each chromosome, we solve the link mapping problem using the link mapping algorithm mentioned in Section 5.3, and calculate the fitness of the entire mapping solution.

The algorithm selects the chromosomes with the top N fitness values to be the parent population for the next iteration after calculating the fitness value of all the chromosomes in the population. The detailed process of FGR-GA is presented in Algorithm 3.

Algorithm 3 FGR-GA algorithm

```
1  Set iteration number $i \leftarrow 0$;
2  Generate the initial population and do feasibility
     checking to obtain $\text{Group}_{\text{init}}$;
3  Do link mapping and evaluate the fitness of each
     chromosome in the population and record the
     solution with the best fitness value $S_{\text{best}}$;
4  Select parent chromosomes according to the fitness
     metric, conduct crossover and mutation operation
     and check feasibility;
5  while $i<I$ do
6      Map virtual links and evaluate the fitness value of
       each chromosome in the population, and record
       the solution with the best fitness value $S'_{\text{best}}$;
7      if $S'_{\text{best}}$ has a higher fitness value than $S_{\text{best}}$
8          $S_{\text{best}} \leftarrow S'_{\text{best}}$;
9      end if
10     $i$++;
11 end while
12 return $S_{\text{best}}$;
```

In Algorithm 3, the inputs are FGR topology $G_r(V_r, E_r)$, substrate network $G_s(V_s, E_s)$, population size M, maximum iteration count I, crossover probability P_c, and mutation probability P_m. The output is the FG embedding solution.

7 Performance evaluation

In this section, we describe a series of simulations used to evaluate the performance of FGR-GA and RRC-FGE. The experimental results show that, compared with the existing algorithms, the two algorithms proposed in this study increase the acceptance ratio and decrease the cost of mapping each forwarding graph request. We also compared the acceptance ratio of FGR-GA under different crossover and mutation probabilities to find the most appropriate system parameter.

7.1 Simulation settings

7.1.1 Topology generation

We used the GT-ITM tool (Wang *et al.*, 2014) to generate a substrate network with 100 nodes and 2000 FGRs. The GT-ITM tool has often been used in practical network topology generation. The substrate nodes were connected randomly with a probability of 0.5; the values of CPU and bandwidth resource were uniformly distributed between 50 and 100. The number of virtual nodes in each virtual network was uniformly distributed from 2 to 10, and the values of the CPU and bandwidth constraints followed a uniform distribution from 0 to 30. In our simulation, each virtual node had a network function type randomly chosen from a total number of 10 network function types; we set the performance interference probability of any two network function types as 0.1. The FG requests arrived according to a Poisson process with 20 requests per 100 time units on average. Each request had an exponentially distributed lifetime with 1000 time units on average.

7.1.2 Parameter setup

To calculate the long-term average revenue, we set w_c and w_b to 3 and 4, respectively. The baseline energy consumption of the server P_b was set to 2, and the energy consumption of each unit of CPU resource P_a was 1. In RRC-FGE, ranking weight σ was set to 0.5, and the available CPU weight d in the resource factor was set to 0.7. In FGR-GA, the population size was set to 40 and the iteration number to 50. When comparing FGR-GA with other embedding algorithms, parameters P_c and P_m were set to 0.8 and 0.1, respectively. In the second step of our experiment, P_c was set to 0.5, 0.7, 0.8, and 0.9 with fixed P_m (0.1), and P_m was set to 0.1, 0.2, 0.3, and 0.4 with fixed P_c to compare the acceptance ratio under different values of P_c and P_m, respectively.

7.1.3 Comparison setup

We implemented the proposed algorithms in C++ and compared our algorithms with two previous algorithms. The notations that we used to refer to the different algorithms are shown in Table 2. The two baseline algorithms were slightly modified to be able to satisfy the performance interference constraint. We used the performance metrics presented in Section 3 to evaluate the performance of these algorithms. All these algorithms were executed on a computer equipped with an Intel® Core^{TM} i7 CPU processor running at 2.67 GHz with two cores, and a RAM of 2 GB.

Table 2 Summary of the compared algorithms

Notation of the algorithm	Algorithm description
BL-SP (Razzaq and Siraj Rathore, 2010)	Baseline algorithm with greedy node mapping and shortest path link mapping
RW-MM-SP (Cheng *et al.*, 2011)	Use a Markov random walk model to rank nodes in the node-mapping stage and the shortest path in the link-mapping stage
RRC-FGE	Our proposed heuristic algorithm using the regional resource clustering metric in the node-mapping stage and the shortest path in the link-mapping stage
FGR-GA	Our proposed genetic algorithm based embedding algorithm

7.2 Evaluation results

1. Our proposed algorithms outperformed baseline algorithms based on the acceptance ratio and average revenue.

Fig. 4 shows that our algorithms achieved a higher acceptance ratio than baseline algorithms in the simulation. This is because our algorithms make better use of topology information, and hence reduce the length of substrate paths that carry virtual links. Also, our algorithms allow multiple virtual nodes within the same request to co-locate on the same substrate node as long as they do not have performance interference. This also reduces bandwidth resource usage in the substrate network. Fig. 5 shows that our proposed algorithms also produced higher average revenue.

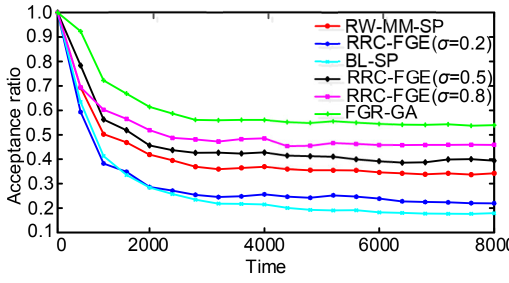

Fig. 4 Acceptance ratio of the compared algorithms

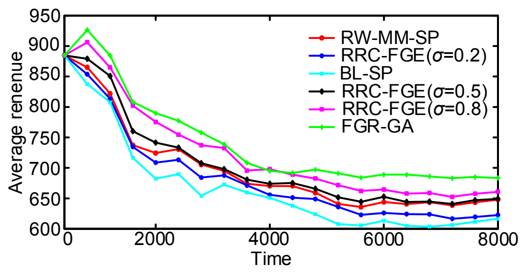

Fig. 5 Average revenue of the compared algorithms

Fig. 4 shows that the heuristic algorithm RRC-FGE achieved a better acceptance ratio with larger σ. The reason is that the heuristic algorithm focuses more on the topology information and makes better use of substrate resources with a larger σ. By using a regional resource clustering metric, RRC-FGE tends to map virtual nodes to the substrate nodes within a small district and hence decreases bandwidth usage for mapping virtual links.

2. The proposed algorithms led to a lower embedding cost and a higher R/C ratio.

As our proposed algorithms tend to map virtual nodes to smaller districts in substrate networks, they reduce bandwidth resource usage by mapping virtual links to shorter substrate paths and decrease deployment costs. Our algorithms also take into account the operation cost of substrate nodes and avoid turning on new substrate nodes from the 'off' state if there are available 'on' state substrate nodes, thereby decreasing operation costs. Our algorithms led to a lower embedding cost and a higher R/C ratio (Figs. 6 and 7).

3. The acceptance ratio of the genetic algorithm based embedding algorithm FGR-GA was affected by system parameters P_c and P_m.

In Figs. 8 and 9, we present the acceptance ratio of our genetic algorithm based embedding algorithm FGR-GA with different crossover and mutation probabilities, respectively. The acceptance ratio

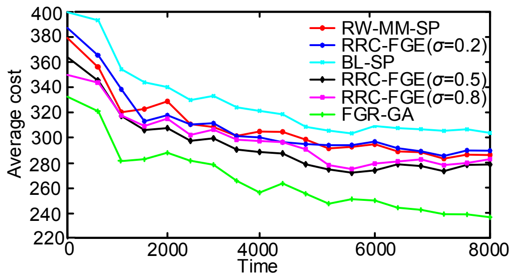

Fig. 6 Average cost of the compared embedding algorithms

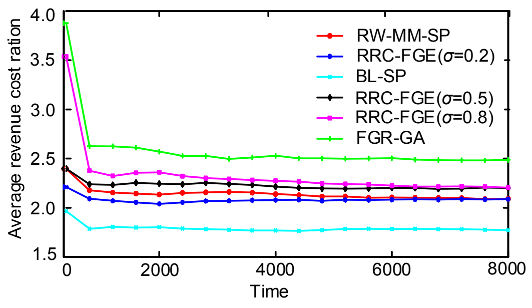

Fig. 7 Average revenue cost ratio of the compared embedding algorithms

increased as the iteration number increased, but when the iteration number exceeded 60, the acceptance ratio tended to be stable. The acceptance ratio of FGR-GA was affected by crossover probability P_c and mutation probability P_m. The crossover probability indicates the percentage of the selected chromosomes whose crossover points are interchanged to generate child chromosomes. A higher crossover probability leads to more child chromosomes which differ from their parents. By introducing this parameter, the algorithm tries to avoid local optimum solutions. However, if the crossover probability is too high, child chromosomes may lose the good performance of their parents. The algorithm with a crossover probability of 0.1 had the highest acceptance ratio, and the acceptance ratio decreased as the crossover probability increased. Similar to the crossover probability, the mutation probability offers another way to avoid local optimum solutions. This parameter indicates the percentage of the child chromosomes whose genes are changed by accident. Again, this parameter should be chosen carefully. According to Fig. 9, the acceptance ratio increased to its maximum as the mutation probability increased to 0.8, and then decreased as the mutation probability increased further. In this case, finding proper system parameters is of great importance.

4. RRC-FGE needs a lower average execution time than FGR-GA.

We compared the average execution time of four algorithms. Although FGR-GA achieved a better average acceptance ratio, it had the highest average execution time compared to the other algorithms (Fig. 10). The reason is that genetics based algorithms generate more chromosomes and select qualified ones. Crossover and mutation operations also need some time. BL-SP had the lowest average execution time, followed by RRC-FGE.

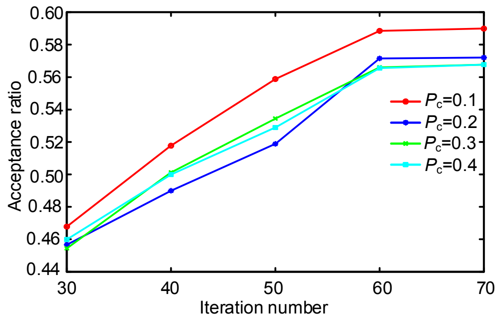

Fig. 8 Acceptance ratios of FGR-GA with different P_c and iteration numbers

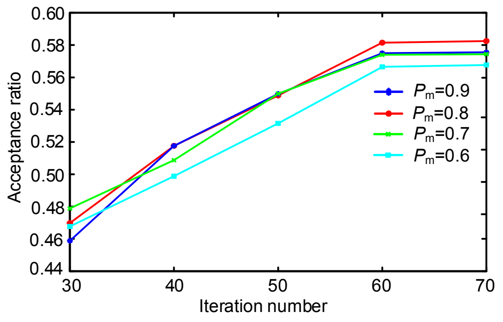

Fig. 9 Acceptance ratios of FGR-GA with different P_m and iteration numbers

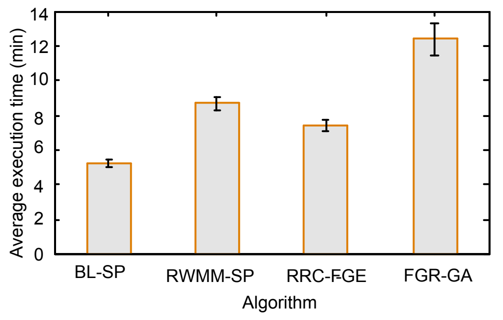

Fig. 10 Average execution time comparison of 2000 forwarding graph regions

8 Conclusions

The FGE problem is a critical problem in network function virtualization. In this study, we study this problem to increase service provider's revenue. First, we formulate the FGE problem as a BIP problem and take network resource constraints into consideration. To reduce the difficulty of solving

NP-hard FGE problem, a regional resource clustering metrics-based topology-aware heuristic embedding algorithm is proposed. To further increase the acceptance ratio and revenue of the service provider, we also propose a genetic algorithm based heuristic embedding algorithm. Simulation results show that our algorithms significantly improve the acceptance ratio and average revenue, and reduce the average cost.

References

Bellavista, P., Callegati, F., Cerroni, W., *et al*., 2015. Virtual network function embedding in real cloud environments. *Comput. Netw.*, **93**(3):506-517. https://doi.org/10.1016/j.comnet.2015.09.034

Bhatia, S., Motiwala, M., Muhlbauer, W., *et al*., 2008. Hosting Virtual Networks on Commodity Hardware. Technical Report, No. GT-CS-07-10. Georgia Institute of Technology, PA.

Chen, D., Lu, L., Shang, M., *et al*., 2012. Identifying influential nodes in complex networks. *Phys. A.*, **391**(4):1777-1787. https://doi.org/10.1016/j.physa.2011.09.017

Cheng, X., Su, S., Zhang, Z., *et al*., 2011. Virtual network embedding through topology-aware node ranking. *ACM SIGCOMM Comput. Commun. Rev.*, **41**(2):38-47. https://doi.org/10.1145/1971162.1971168

Chowdhury, M., Rahman, M., Boutaba, R., *et al*., 2009. Vineyard: virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Trans. Netw.*, **20**(1):206-219. https://doi.org/10.1109/TNET.2011.2159308

Eppstein, D., 1998. Finding the *K* shortest paths. *SIAM J. Comput.*, **28**(2):652-673. https://doi.org/10.1137/S0097539795290477

ETSI NFVISG (European Telecommunications Standards Institute Network Functions Virtualization Industry Specification Group), 2013. Network Functions Virtualization, White Paper. http://www.esti.org/technologies cluster/technologies/nfv [Accessed on Nov. 2, 2016].

Fagiolo, G., 2007. Clustering in complex directed networks. *Phys. Rev. E*, **76**(2):12-23. https://doi.org/10.1103/PhysRevE.76.026107

Fan, J., Ammar, M., 2006. Dynamic topology configuration in service overlay networks: a study of reconfiguration policies. Proc. 25th IEEE Int. Conf. on Computer Communications, p.21-32. https://doi.org/10.1109/INFOCOM.2006.139

Goldberg, D., 2009. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Professional, USA, p.216-236.

Gong, L., Wen, Y., Zhu, Z., *et al*., 2014. Toward profit-seeking virtual network embedding algorithm via global resource capacity. Proc. IEEE Int. Conf. on Computer Communications, p.1-9. https://doi.org/10.1109/INFOCOM.2014.6847918

Koh, Y., Knauerhase, R., Brett, P., *et al*., 2007. An analysis of performance interference effects in virtual environments. Proc. IEEE Int. Symp. on Performance Analysis of Systems & Software, p.200-209. https://doi.org/10.1109/ISPASS.2007.363750

Lischka, J., Karl, H., 2009. A virtual network mapping algorithm based on subgraph isomorphism detection. Proc. 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures, p.81-88. https://doi.org/10.1145/1592648.1592662

Lu, J., Turner, J., 2006. Efficient Mapping of Virtual Networks onto a Shared Substrate. Technical Report No. WUCSE-2006-35, Washington University in St Louis, Washington.

Mehraghdam, S., Keller, M., Karl, H., *et al*., 2014. Specifying and placing chains of virtual network functions. Proc. 3rd IEEE Int. Conf. on Cloud Networking, p.7-13. https://doi.org/10.1109/CloudNet.2014.6968961

Mei, Y., Liu, L., Pu, X., *et al*., 2013. Performance analysis of network I/O workloads in virtualized data centers. *IEEE Trans. Serv. Comput.*, **6**(1):48-63. https://doi.org/10.1109/TSC.2011.36

Pu, X., Liu, L., Mei, Y., *et al*., 2010. Understanding performance interference of I/O workload in virtualized cloud environment. Proc. 3rd IEEE Int. Conf. on Cloud Computing, p.51-58. https://doi.org/10.1109/CLOUD.2010.65

Razzaq, A., Siraj Rathore, M., 2010. An approach towards resource efficient virtual network embedding. Proc. 2nd Int. Conf. on Evolving Internet, p.68-73. https://doi.org/10.1109/INTERNET.2010.21

Shea, R., Wang, F., Wang, H., *et al*., 2014. A deep investigation into network performance in virtual machine based cloud environments. Proc. IEEE Int. Conf. on Computer Communications, p.1285-1293. https://doi.org/10.1109/INFOCOM.2014.6848061

Su, S., Zhang, Z., Cheng, X., *et al*., 2012. Energy-aware virtual network embedding through consolidation. Proc. IEEE Int. Conf. on Computer Communications Workshops, p.127-132. https://doi.org/10.1109/INFCOMW.2012.6193473

Wang, Z., Wu, J., Wang, Y., *et al*., 2014. Survivable virtual network mapping using optimal backup topology in virtualized SDN. *China Commun.*, **11**(2):26-37. https://doi.org/10.1109/CC.2014.6821735

Xia, S., Zhang, Y., Green, H., *et al*., 2014. Network function placement for NFV chaining in packet/optical data centers. Proc. European Conf. on Optical Communication, p.1-3. https://doi.org/10.1109/ECOC.2014.6963925

Yu, M., Yi, Y., Rexford, J., *et al*., 2008. Rethinking virtual network embedding: substrate support for path splitting and migration. *ACM SIGCOMM Comput. Commun. Rev.*, **32**(2):17-29. https://doi.org/10.1145/1355734.1355737

Zhang, S., Qian, Z., Wu, J., *et al*., 2012. An opportunistic resource sharing and topology-aware mapping framework for virtual networks. Proc. IEEE Int. Conf. on Computer Communications, p.2408-2416. https://doi.org/10.1109/INFCOM.2012.6195630