Topology-aware Virtual Network Embedding Using Multiple Characteristics

Jianxin Liao[1*], Min Feng[1], Tonghong Li[2], Jingyu Wang[1], Sude Qing[1]
[1]State Key Lab of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China
[2]Department of Computer Science, Technical University of Madrid, Madrid, Spain
[Email: [1]{liaojx, fengmin, wangjingyu, qingsd}@bupt.edu.cn; [2]tonghong@fi.upm.es; [*]jianxin.liao@sohu.com]
Corresponding author: Jianxin Liao

Received July 13, 2013; revised November 22, 2013; accepted December 28, 2013; published January 29, 2014

Abstract

Network virtualization provides a promising tool to allow multiple heterogeneous virtual networks to run on a shared substrate network simultaneously. A long-standing challenge in network virtualization is the Virtual Network Embedding (VNE) problem: how to embed virtual networks onto specific physical nodes and links in the substrate network effectively. Recent research presents several heuristic algorithms that only consider single topological attribute of networks, which may lead to decreased utilization of resources. In this paper, we introduce six complementary characteristics that reflect different topological attributes, and propose three topology-aware VNE algorithms by leveraging the respective advantages of different characteristics. In addition, a new KS-core decomposition algorithm based on two characteristics is devised to better disentangle the hierarchical topological structure of virtual networks. Due to the overall consideration of topological attributes of substrate and virtual networks by using multiple characteristics, our study better coordinates node and link embedding. Extensive simulations demonstrate that our proposed algorithms improve the long-term average revenue, acceptance ratio, and revenue/cost ratio compared to previous algorithms.

Keywords: Network Virtualization, Virtual Network Embedding, Topological Characteristics, Node Ranking, Topology Decomposition

This work was jointly supported by: (1) the National Basic Research Program of China (No. 2013CB329102); (2) National Natural Science Foundation of China (No. 61372120，61271019, 61101119, 61121001, 61072057, 60902051); (3) PCSIRT (No. IRT1049); (4) Beijing Higher Education Young Elite Teacher Project (Grant Nos. YETP0473); (5) MICINN (No.TIN2010-19077); (6) CAM (No.S2009TIC-1692)

http://dx.doi.org/10.3837/tiis.2014.01.009

1. Introduction

As a powerful way to eradicate the ossifying forces of the Internet, network virtualization is an important technique for designing the future Internet architecture, and has obtained wide attention in recent years [1-5]. Network virtualization allows multiple heterogeneous virtual networks (VNs), each customized to a specific topology and purpose, to run on a shared substrate network simultaneously. It will benefit many new computing and networking paradigms, such as Cloud Computing platforms, Data Center networks, and Software Defined Networking (SDN).

In network virtualization environment, traditional Internet Service Providers (ISPs) nowadays are decoupled into two independent entities: Infrastructure Providers (InPs) who deploy and manage physical infrastructures, and Service Providers (SPs) who rent sliceable physical resources from InPs to build VNs and offer diverse services to end users. A VN has a customized logical topology that is composed of virtual nodes and links with various resource requirements. To serve as many different virtual network requests (VNRs) as possible and profit from them, InPs strive to make efficient use of physical resources by optimizing mapping algorithms that map/embed each VN onto specific physical nodes and links in the substrate network. (The words "embed" and "map" will not be differentiated in this paper.) This is known as the Virtual Network Embedding (VNE) problem. However, the VNE problem is NP-hard whether the problem space is restricted or not. Computational intractability urges researchers to devise various heuristic algorithms to find practical solutions [6-13].

The VNE problem can be divided into two sub-problems: the node mapping where virtual nodes are allocated in physical nodes and the link mapping where virtual links connecting these virtual nodes are mapped to physical paths connecting the corresponding physical nodes. Most heuristic algorithms employ the "greedy" node mapping (e.g., [7, 8, 9, 11, 12, 13]), also called L2S2 mapping (which stands for "large-to-large and small-to-small" mapping [11]). It maps the virtual nodes requiring more resources onto the physical nodes with more resources, so as to minimize the use of resources at bottleneck nodes and links. This greedy way helps to satisfy the resource requirements of current VNR and balance the loads of the substrate network. Furthermore, it is beneficial to future VNRs that require specific physical nodes and links with scarce resources.

The most important part in the "greedy" node mapping is the node ranking, i.e., how to measure every node in a substrate or virtual network, in order to correspondingly match and embed them one after another. Recent algorithms measure a node in the substrate network or a VN quantitatively by using its CPU and bandwidth (e.g., [9, 12]), or certain network topological attribute (e.g., [11, 13]). However, topological attributes could have significant effect on the performance of embedding algorithms. The existing algorithms, which typically ignore the topological structure or only consider single topological attribute in node ranking, barely coordinate node and link mapping, and may lead to decreased utilization of the substrate network resources, meaning low revenues or high costs for InPs. This calls for the design of new algorithms that synthetically measure a node's resources and topological attributes at the same time.

Motivated by the disadvantages of traditional greedy node mapping, we present a new topology-aware approach in this paper, which utilizes different network topological attributes simultaneously. Topological attributes emerge in the form of diverse characteristics, which measure the relative influence or importance of nodes and links from different aspects. Six characteristics reflecting different topological attributes are introduced: "degree", "strength",

"closeness (farness) centrality", "betweenness centrality", "eigenvector centrality" and "Katz centrality". Three topology-aware heuristic algorithms with multiple characteristics are proposed accordingly, to leverage the respective advantages of different characteristics. Our algorithms consider not only the resource requirements of nodes but also their topological attributes, which should better coordinate node mapping and link mapping.

In most cases, nodes in a VN play different roles according to the services they provide, such as directories, file servers, terminals, etc. Even if all the nodes in some specific VNs are considered to be equivalent, they can be differentiated hierarchically according to their abilities or actual effects in the network. Therefore in the VNE problem, using topology decomposition to disentangle the hierarchical structure of a VN can make the mapping process easier. Inspired by the K-core decomposition algorithm presented in [12], we come up with a new KS-core decomposition algorithm based on two characteristics, "degree" and "strength", to better disentangle the hierarchical topological structure of a VN and divide it into a core network and multiple edge networks. The KS-core decomposition algorithm is an example of utilizing topological characteristics from a different aspect in the VNE problem.

Due to the overall consideration of topological attributes of substrate and virtual networks by using multiple characteristics, our new topology-aware approaches can better coordinate node and link mapping. Extensive simulations demonstrate that our approaches improve the long-term average revenue, acceptance ratio, and revenue/cost ratio compared to previous algorithms.

In summary, this paper presents the following major contributions:

a) We introduce six topological characteristics, which represent different topological attributes, in the design of topology-aware VNE algorithms. To the best of our knowledge, most of these characteristics are first applied to the VNE problem.
b) We propose three topology-aware VNE algorithms based on multiple topological characteristics, and three correspondingly modified algorithms for the algorithm presented in [11]. Equations of node ranking are well designed to leverage the respective advantages of different characteristics.
c) The KS-core decomposition algorithm based on the characteristic "degree" and "strength" is devised to better disentangle the hierarchical structure of each VN.
d) Extensive simulations are conducted to compare 7 proposed algorithms with 4 previous algorithms accordingly. The results demonstrate that our algorithms significantly increase the long-term average revenue, acceptance ratio, and revenue/cost ratio.

The remainder of this paper is organized as follows. Section 2 discusses some previous VNE algorithms related to our work, and Section 3 presents the general model of the VNE problem, along with three objectives used in the evaluation. Six topological characteristics are introduced in Section 4, which are used to design three new algorithms and three modified algorithms in Section 5. We devise the KS-core decomposition algorithm in Section 6. Section 7 presents the simulation results and the paper is concluded in Section 8.

2. Related Work

Previous research presents some heuristic algorithms that consider the network topologies more or less. Szeto et al. [6] propose a link mapping scheme based on Multi-Commodity Flow (MCF) problem. In their scheme, nodes in the substrate network are classified as "the access nodes" and "the transit nodes", thus each pair of access nodes is treated as a commodity and a set of resources are pre-allocated to them.

To utilize the flexibility provided by small topologies, Zhu and Ammar [7] split VNRs into many star sub-networks. During the node mapping process, the center node with highest degree in a star sub-network is mapped to the physical node with lowest CPU and bandwidth occupancy rate. Once the center node has been determined, other nodes in the same sub-network with higher degrees are embedded onto physical nodes with lower resource occupancy rates and shorter distances to the center, one by one. However, their algorithm assumes that substrate resources are unlimited, and focuses on load balance in the substrate network without the need of admission control. Besides, not all the topologies can be appropriately divided into star sub-networks. Houidi et al. [8] adopt similar idea of topology splitting in [7]. Every center node is mapped onto the physical node with maximum resources, while every edge node is mapped onto the physical node with shortest path to the center one. As the centralized VNE algorithms suffer from the scalability limitation and high latency, they present a distributed VNE algorithm by using a multi-agent system. The distributed algorithm can improve the system's robustness, reduce the communication costs and support the high-speed parallel computing. Nevertheless, it has relatively poor performance although unlimited resources are assumed to accept all the VNRs.

Yu et al. [9] assume that the substrate network supports path splitting and path migration, rather than restricting the problem space as many previous algorithms did. Flexible splitting of virtual links over multiple physical paths and periodically re-optimizing the embedding of existing virtual links allow the substrate network to accept more VNRs, and thus increase the average revenue of InPs. A "greedy" node mapping algorithm similar to [7] is proposed, in which the product of CPU and adjacent bandwidth is newly defined for node ranking. However, a virtual node may choose a physical node with more than adequate CPU resource but not enough bandwidth resource, leading to the failure of subsequent link mapping process.

Chowdhury et al. [10] introduce the correlation between node mapping and link mapping. They formulate the VNE problem as a Mixed Integer Programming problem by constructing an augmented substrate network based on the location constraints of virtual nodes. To decide which physical nodes should be chosen, they relax the integer constraints and use deterministic or randomized rounding techniques. After the node mapping process, MCF algorithm or the shortest path algorithm is used to map the virtual links. However, the results from the node mapping process may result in an infeasible link mapping.

Cheng et al. [11] use Google's successful PageRank algorithm in the web search domain for reference, by applying Markov's Random Walk model in node ranking. Every node in the substrate network and VNs is ranked based on its resources (CPU and bandwidth, similar to [9]), as well as the ranks of its connected nodes. The nodes are mapped in a greedy way according to their ranking values.

Qing et al. [12] advocate a hybrid VNE algorithm by pruning the topologies of VNs with the K-core decomposition. A VN is differentiated as a core network and multiple edge networks, which are respectively applied with certain two-stage algorithm and one-stage algorithm to leverage the respective advantages of these two kinds of algorithms simultaneously. In the core network and edge networks, node mapping employs the same "greedy" algorithm as previous algorithms did. Nevertheless, the K-core decomposition is not adequate to the weighted networks and some specific topologies.

Wang et al. [13] exploit the "closeness centrality" (which will be discussed in details in Section 4) and propose two VNE algorithms. The node mapping process also works in a greedy way, where node ranking is based on each node's closeness centrality. However, the closeness centrality is one network topological attribute and only measures how close a node is to other nodes. Their classical closeness algorithm has high acceptance ratio but low

revenue/cost ratio. Thus, more topological attributes should be taken into consideration in order to achieve the optimal performance of VNE.

3. Virtual Network Embedding Problem

In this section, we describe the general model and objectives of the VNE problem.

Substrate Network The topology of the substrate network is not always clear, especially when multiple InPs are involved in the multi-domain VNE problem. By applying the topology discovery tool in [14], the topology of the substrate network can be well revealed. Similar to previous work in [9, 11], a substrate network is denoted by a weighted undirected graph $G^s = (N^s, L^s, A_N^s, A_L^s)$, where N^s refers to the set of physical nodes and L^s refers to the set of physical links. A_N^s and A_L^s denote the attributes of the physical nodes and links respectively. The attributes of a node include computation capacity, memory, disk capacity, buffer size, geographical location, and so on [15]. And the attributes of a link include bandwidth, propagation delay, bit error rate, and so on. In this paper we consider the typical CPU capacity as the node attribute and the typical bandwidth as the link attribute, just the same as most previous algorithms did. Therefore, each physical node $n^s \in N^s$ is associated with an available CPU capacity $a_n^s \in A_N^s$, while each physical link $l^s \in L^s$ is associated with an available bandwidth capacity $a_l^s \in A_L^s$. In addition, we use P^s to denote the set of all loop-free physical paths in the substrate network.

The right side of **Fig. 1** demonstrates a substrate network, where the numbers next to nodes are their available CPU resources and the numbers in parentheses over links represent their available bandwidths. The links thicker are the links with more bandwidth resources.

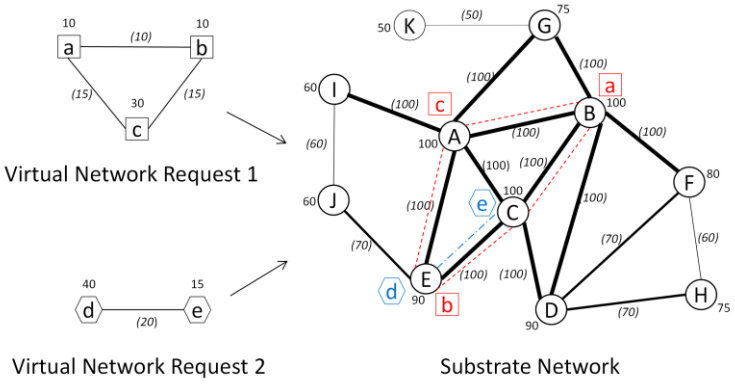


Fig. 1. An example of virtual network embedding

Virtual Network Request We use another weighted undirected graph $G^v = (N^v, L^v, C_N^v, C_L^v)$ to denote a VN. N^v represents the set of virtual nodes and L^v represents the set of virtual links. Virtual nodes and links have CPU and bandwidth constraints, which are respectively denoted by C_N^v and C_L^v.

We also denote the i^{th} VNR by $VNR_i(G^v, t, d, t_d)$. t is the arrival time of the VNR, d is its duration, i.e., for how long the VN plans to rent the resources in the substrate network, and t_d is its delay, i.e., for how long the VN is willing to wait before it is successfully embedded. Once VNR_i arrives, the substrate network tries to assign resources to this VNR to meet the resource requirements of its virtual nodes and links. If there are not sufficient substrate resources available, VNR_i will be postponed. If a VNR could not be embedded after time t_d, it will be rejected.

The left side of **Fig. 1** depicts two VNRs: VNR_1 requires 10 units of CPU resources each at node *a*, *b,* and 30 at node *c*, and requires 15 units of bandwidth resources each over link (*a*, *c*), (*b*, *c*), and 10 over link (*a*, *b*). VNR_2 needs a link of 20 units of bandwidth to connect node *d* and *e*, whose CPU constraints are 40 units and 15 units respectively.

VNE Problem The embedding of a VNR onto the substrate network is defined by the mapping M from G^v to a subset of G^s:

$$M\colon (N^v, L^v, C_N^v, C_L^v) \rightarrow (N', P', A_N', A_L')$$

where $N' \subset N^s$, $P' \subset P^s$, and A_N', A_L' are the node and link resources assigned to the VNR, with all its constraints satisfied.

Fig. 1 also shows the VNE solutions for VNR_1 and VNR_2. For VNR_1, the virtual node *a*, *b*, and *c* are mapped to the physical node *B*, *E*, and *A*, and the virtual link (*a*, *b*), (*a*, *c*), and (*b*, *c*) are mapped to the physical path (*B*, *C*, *E*), (*B*, *A*), and (*E*, *A*). VNR_2 is mapped likewise. Note that the virtual nodes from different VNRs can be mapped onto the same physical node, but different virtual nodes from the same VNR cannot.

If a VNR has been successfully embedded, the assigned physical resources will be dedicated to it during its duration. When the time of the VN is over, the assigned resources will be released, and get ready to be reassigned for other VNRs.

Objectives VNE is a multi-objective optimization problem. From InPs' perspective, the VNE problem has three main objectives: maximum revenue, maximum acceptance ratio, and maximum revenue/cost (RC) ratio. Revenue always comes first in business, and acceptance ratio ensures it. High revenue and acceptance ratio mean not only high market share but also a good reputation. Once the revenue is guaranteed, the efficiency of the resource utilization of the substrate network becomes important, which is measured by RC ratio.

Similar to previous work in [7, 9], the revenue of maintaining an embedded VNR at time t is defined as the weighted sum of total resources it demands:

$$R(G^v, t) = \sum_{n^v \in N^v} CPU(n^v) + \alpha \sum_{l^v \in L^v} BW(l^v) \tag{1}$$

where $CPU(n^v)$ and $BW(l^v)$ are CPU constraint of virtual node n^v and bandwidth constraint of virtual link l^v, respectively. α is a coefficient used to balance the relative revenues from CPU and bandwidth. So our first objective, the long-term average revenue, can be formulated as follows:

$$\lim_{T \rightarrow \infty} \frac{\sum_{t=0}^{T} R(G^v, t)}{T} \tag{2}$$

Let VNR^s denote the total VNRs and VNR^m denote the number of successfully mapped VNRs. Our second objective, the long-term acceptance ratio is defined as:

$$\lim_{T \rightarrow \infty} \frac{\sum_{t=0}^{T} VNR^m}{\sum_{t=0}^{T} VNR^s} \tag{3}$$

A little different from the revenue, the cost of mapping a VNR at time t is defined as the total consumption of resources:

$$C(G^v, t) = \sum_{n^v \in N^v} CPU(n^v) + \alpha \sum_{p' \in P'} BW(p') \times hop(p') \tag{4}$$

where P' is the set of physical paths assigned to the virtual links, $hop(p')$ is the number of hops in physical path p', and $BW(p')$ is the dedicated bandwidth over p'. Note that the revenue of a VNR is not affected by the mapped physical paths. But when calculating the cost, InPs have to count the number of hops in each mapped physical path. This is reasonable, because obviously SPs do not want to pay for more number of hops assigned to the same virtual links.

With the revenue and cost of a VNR, we present the long-term RC ratio, in which we can see that to maximize the RC ratio, the number of hops of each mapped physical path should be minimized:

$$\lim_{T\to\infty} \frac{\sum_{t=0}^{T} R(G^v, t)}{\sum_{t=0}^{T} C(G^v, t)} \tag{5}$$

4. Topological Characteristics

Network topological attributes have a critical effect on the performance of VNE algorithms, and they emerge in the form of diverse characteristics, every one of which measures the relative influence or importance of nodes and links from different aspects. Degree is one basic topological characteristic of a node. So is the total bandwidth of all adjacent links of a node, named "strength". Other topological characteristics can be obtained through centrality measures in the network analysis, which is widely used in disciplines like sociology [16]. In the scope of network analysis, nodes are characterized from multidimensional measures, including closeness centrality, betweenness centrality, eigenvector centrality, and Katz centrality [17, 18].

The degree of a node is the number of immediate links to its neighbors. Degree can be interpreted as the immediate possibility that a node contacts with others. The degree of a node n_i can be denoted as:

$$D(n_i) = deg(n_i) \tag{6}$$

In a weighted network, links can be treated more than binary interactions. So the strength comes, which takes into account the total bandwidth of the same adjacent links contributing to the degree. The strength of node n_i is defined as follows, where $L(n_i)$ is the set of adjacent links of n_i:

$$S(n_i) = \sum_{l\in L(n_i)} BW(l) \tag{7}$$

The closeness centrality is a natural distance metric between all pairs of nodes, based on the length of their shortest paths. It can be regarded as a measure of how close a node is to the network center. The farness of a node is defined as the sum of its shortest distances to all the other nodes, and its closeness is defined as the reciprocal of farness. So the lower the farness of a node is, the higher its closeness is, and the more central it is. The farness of node n_i is defines as:

$$F(n_i) = \sum_{j=1}^{N} d(n_i, n_j) \tag{8}$$

where $d(n_i, n_j)$ is the length of shortest path between node n_i and n_j, and N is the sum of nodes. Note that $d(n_i, n_j) = 0$ when $i = j$. Similar to [13], the closeness of n_i is defined as:

$$C(n_i) = \frac{1}{\sum_{j=1}^{N} d(n_i, n_j)} \tag{9}$$

The betweenness centrality quantifies the number of times a node acts as a bridge along the shortest path between any other two nodes. It reflects the switching capacity of a node in the network and has been applied to control the traffic flow in congestion. The betweenness of node n_i is denoted by:

$$B(n_i) = \sum_{i\neq j\neq k} \frac{P_{n_j,n_k}(n_i)}{P_{n_j,n_k}} \tag{10}$$

where P_{n_j,n_k} is the number of shortest paths between node n_j and n_k, and $P_{n_j,n_k}(n_i)$ is the number of those passing through n_i.

The eigenvector centrality uses eigenvectors of the adjacency matrix corresponding to a network, to determine the nodes that tend to be frequently visited. It assigns a relative score to every node based on the concept that connections to high-score nodes contribute more to the score of the node in question. The eigenvector centrality is mainly used to measure the directed networks. For a given graph $G = (N, L)$, the adjacency matrix is defined to be $A = (a_{n_i,n_j})$. $a_{n_i,n_j} = 1$ if node n_i links to n_j, and $a_{n_i,n_j} = 0$ otherwise. Let γ denote the eigenvalue, the eigenvector centrality of node n_i is formulated by:

$$E(n_i) = \frac{1}{\gamma} \sum_{n_j \in N} a_{n_i,n_j} E(n_j) \tag{11}$$

The Katz centrality can be viewed as an extension of the eigenvector centrality. The relative influence of a node within a network is measured by summing the weighted paths between this node and all reachable nodes in the network. Paths connecting the node with its immediate neighbors carry higher weights than those connecting it with other nodes far away. Similar to the eigenvector centrality, the Katz centrality is also mainly used in directed networks, and is more suitable in directed acyclic graphs where the eigenvector centrality is rendered useless [16]. Similar to the eigenvector centrality, the Katz centrality is formulated as follows:

$$K(n_i) = \mu \sum_{n_j \in N} a_{n_i,n_j} \left[K(n_j) + 1\right] \tag{12}$$

where μ is an attenuation factor by which the contribution of a distant node is penalized.

To summarize, a node with high degree has many connections while a node with high strength has strong connections. A node with high closeness centrality is, on average, at a short distance from other nodes in the network. A node with high betweenness centrality lies on many of the shortest paths between any other two nodes in the network [19]. A node with high eigenvector centrality or Katz centrality has a high probability of connecting to other nodes with high eigenvector centrality or Katz centrality. All these characteristics measure a node's relative influence or importance within the network from different aspects.

5. Topology-aware Mapping Algorithms With Multiple Characteristics

Since Yu et al. [9] use the product of a node's CPU and adjacent bandwidth in node ranking, many greedy node mapping algorithms later adopt the similar method. However, these node mapping algorithms could not give every node a comprehensive rank. A virtual node may choose a physical node with more than adequate CPU resource but not enough bandwidth resource, leading to the failure of subsequent link mapping process. Even if the bandwidth resource is adequate, adjacent virtual nodes may be mapped onto physical nodes far away from each other, which will cause higher costs and increase the network fragmentation that gradually prevents the substrate network from accepting more VNRs.

We argue that the disadvantages of traditional greedy node mapping can be eliminated by utilizing more topological attributes. Therefore, we propose a topology-aware approach that integrates multiple topological characteristics into the node mapping process and leverages their respective advantages. In addition, our approach aims to better coordinate node and link mapping, and thus improve the success rate and efficiency of the link mapping process and the performance of VNE.

A motivational example of using the characteristic "degree" in node ranking is illustrated in **Fig. 2**, where the numbers next to nodes are their CPU capacities and the numbers in parentheses over links represent their bandwidth capacities. The links thicker are the links with more bandwidths. Node N and M have equal CPU capacities (i.e., 75) and strengths (i.e., 340), and are considered equivalent in traditional node ranking. However, Node N should have higher priority than Node M, especially when path splitting is supported by the substrate network. This is because the degree of Node N is 5, higher than the degree of Node M (i.e., 4), which means that Node N has not only strong connections but also more connections. Therefore, mapping a virtual node to Node N may have a higher chance to achieve a successful and low-cost link mapping.

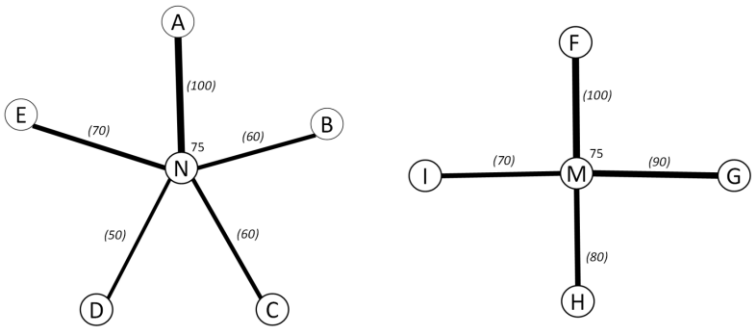


Fig. 2. A motivational example

Similar to "degree", other characteristics, "closeness (farness) centrality", "betweenness centrality", "eigenvector centrality", and "Katz centrality", can also help to improve the performance of VNE. When the characteristics used in node ranking fail to tell the proper differences between nodes, other topological characteristics can be used to rank the nodes more accurately, and thus the success rate and efficiency of subsequent link mapping are improved. Our node ranking reflects not only nodes' CPU resources but also topological attributes. Obviously, using such a node ranking method to construct VNE solutions can increase the possibility of satisfying the resource requirements of VNRs and reduce the costs of embedding and network fragmentation.

Because the effects of these topological characteristics on the performance of VNE are not well understood, we attempt to integrate them into node ranking step by step, so as to better reveal and compare their influence over the practical VNE algorithms. In the remainder of this section, we propose three heuristic algorithms based on "degree", "strength", "closeness (farness) centrality", and "betweenness centrality" in *5.1*, *5.2* and *5.3* from simple to complex. As for *5.4*, we adopt the algorithm in [11], which includes the implementation of "eigenvector centrality" and "Katz centrality", and improve it by applying our three new node ranking methods.

5.1. Mapping Algorithm with Degree

Firstly, we extend the traditional node mapping with "degree". The degree and strength of a node both measure the extent to which it is connected to the rest of the network, from two complementary aspects. By adding degree, we wish to enhance the influence of local

Algorithm 1 node mapping algorithm with degree

1. Collect all the virtual network requests (VNRs) just arrived or should be mapped during current time window.
2. Sort these VNRs according to their revenues.
3. ***if*** no VNR left, ***stop***.
4. Take the unchecked VNR with the largest revenue.
5. Calculate $R_d(n_i)$ in (13) of all the physical nodes with their real-time residual resources.
6. Calculate $R_d(n_i)$ in (13) of all the virtual nodes using node and link constraints.
7. Rank physical nodes and virtual nodes respectively, based on $R_d(n_i)$.
8. ***for*** all the unmapped virtual nodes of the VNR ***do***
9. Choose the virtual node with the highest rank;
10. Mapping it to the physical node with the highest rank that satisfies every constraint and is unmapped with any other nodes from the same VNR.
11. ***if*** fail to embed ***then***
12. Postpone or reject the VNR (VNR checked).
13. ***GOTO*** step 3.
14. ***end if***
15. ***end for***
16. ***GOTO*** step 3.

resources in node mapping. We define the amount of resources for node n_i by:

$$R_d(n_i) = CPU(n_i) \times deg(n_i) \sum_{l \in L(n_i)} BW(l) \qquad (13)$$

Our algorithm is a two-stage VNE algorithm. We describe the node mapping algorithm with degree in Algorithm 1. Time is discretized into consecutive time windows and the VNR queue is applied to store a group of unmapped and incoming VNRs with irregular arrival times during a time window. During the node mapping process, we collect in the queue all the VNRs that just arrive or should be mapped during the current time window at first, and then sort them according to their revenues. The unchecked VNR with largest revenue has priority. When a VNR is to be embedded, the R_d in (13) for each virtual node and physical node will be calculated and act as their ranking values. Note that the CPU, degree and strength of a physical node in R_d are all calculated using its real-time residual resources. Thus the amount of available resources R_d of a physical node can reflect the dynamic state of the substrate network. Our algorithm maps the virtual node with the highest rank to the physical node with the highest rank that satisfies every constraint and is unmapped with any other nodes from the same VNR. Other virtual nodes of the VNR will be mapped in the same greedy way according to the order of their ranks. Some VNRs may be postponed due to the lack of resources in the substrate network, and have to wait for next time window to be mapped. A VNR is rejected once it cannot be served within its tolerant delay. The algorithm will go back to the queue to find the next unchecked VNR with the largest revenue until every VNR is mapped, postponed or rejected.

The link mapping process follows the node mapping process. Virtual links are embedded by the k-shortest path algorithm [20] if path splitting is not supported by the substrate network or the Multi-Commodity Flow (MCF) algorithm [21] if path splitting is supported. The k-shortest path algorithm is shown in Algorithm 2. The MCF algorithm is implemented by the PPRN package [22], which is generally appropriate to solve a high variety of MCF problems with linear/nonlinear objective functions and with/without linear side constraints.

Algorithm 2 link mapping by the k-shortest path algorithm

1. Collect all the virtual network requests (VNRs) that successfully complete the node mapping process during current time window.
2. Sort these VNRs according to their revenues.
3. ***if*** no VNR left, ***stop***.
4. Take the unchecked VNR with the largest revenue.
5. ***for*** every virtual link of the VNR ***do***
6. k=1;
7. ***while*** k < K ***do***
8. Search the k-shortest paths for the virtual link.
9. ***if*** find one physical path with adequate bandwidth ***then***
10. Mapping the virtual link to the physical path.
11. ***break;***
12. ***else*** k++;
13. ***end if***
14. ***end while***
15. ***if*** fail to embed ***then***
16. Postpone or reject the VNR (VNR checked).
17. ***GOTO*** step 3.
18. ***end if***
19. ***end for***
20. ***GOTO*** step 3.

Once all the virtual nodes and links have been mapped, the embedding of a VNR succeeds. The dedicated resources of a VNR will be released when its duration is over and get ready to be reassigned for other VNRs.

5.2. Mapping Algorithm with Degree and Closeness

Secondly, we take into consideration the closeness (farness) centrality. The closeness and farness is a pair of interesting distance metrics. They are analogous to the distance in physics. Let us consider two classical equations in physics: Newton's law of universal gravitation in gravitational field is $F = G\frac{m_1 m_2}{r^2}$, and Coulomb's law in electromagnetism is $|F| = k_e\frac{|q_1 q_2|}{r^2}$. Just like these two equations in physics describing the interactions between discrete particles, the distances between them have an inverse-square effect on their interactions. This is because the interactions in universe usually decrease over distances in an inverse-square way, rather than an isometric way. The closer one has a much stronger effect on the one in question than the farther one. We argue that the above principle applies to the node's farness in the context of node ranking as well, i.e., the farness of a node has an inverse-square effect on its importance in the network. Therefore we formulate the amount of resources for node n_i by:

$$R_{dc}(n_i) = CPU(n_i)\frac{deg(n_i)\sum_{l\in L(n_i)} BW(l)}{\left[\sum_{j=1}^{N} d(n_i, n_j)\right]^2} \tag{14}$$

Steps of the algorithm with degree and closeness (farness) are similar to those of the algorithm with degree in *5.1*. The main difference is the calculating method of node ranking, in which $R_d(n_i)$ is replaced by $R_{dc}(n_i)$. And we employ the Floyd–Warshall algorithm to obtain the closeness (farness) by calculating shortest paths between all pairs of nodes.

5.3. Mapping Algorithm with Degree, Closeness and Betweenness

Thirdly, we further extend the node mapping with betweenness centrality, which captures the switching capacity of a node. A node with higher switching capacity should have higher priority than the one with lower switching capacity. Therefore, we integrate it into node ranking and define the amount of resources for n_i by:

$$R_{dcb}(n_i) = CPU(n_i) \frac{deg(n_i)\left[\sum_{l \in L(n_i)} BW(l)\right]\left[\sum_{i \neq j \neq k} \frac{P_{n_j,n_k}(n_i)}{P_{n_j,n_k}}\right]}{\left[\sum_{j=1}^{N} d(n_i, n_j)\right]^2} \quad (15)$$

The algorithm with degree, closeness and betweenness is similar to the algorithm with degree and closeness in *5.2*. However, the calculating method of node ranking changes to $R_{dcb}(n_i)$. In addition, calculating both the closeness (farness) and the betweenness of each node involves calculating shortest paths between all pairs of nodes. So we modify the Floyd–Warshall algorithm in order to find and locate all the shortest paths between any two nodes.

5.4. Mapping Algorithm with Eigenvector Centrality and Katz Centrality

Finally, we try to implement the eigenvector centrality and the Katz centrality. Google's PageRank is a variant of eigenvector centrality and Katz centrality [23] that assign relative scores to all nodes in the network, and the major difference is the scaling factors. So is the Random Walk (RW) algorithm in [11]. The RW algorithm transplants Google's PageRank algorithm into the VNE problem, along with a Markov Random Walk model.

PageRank considers a link from web page A to web page B as a vote. A web page is considered more important if more important web pages (either in quantity or quality) vote for it. In doing so, the topology of the web can influence the PageRank of a web page. In the context of VNE, the RW algorithm considers a node to be more important if a node links to more nodes with more resources [11]. Treating the connectivity between any two nodes as a Markov chain transition with certain probability, the RW algorithm iteratively calculates the relative resource quality of a node based on the product of its CPU and adjacent bandwidth, and the qualities of its neighbors.

We adopt the RW algorithm, which includes the implementation of eigenvector centrality and Katz centrality. Steps of the RW algorithm will not be repeatedly described here. To further integrate other topological characteristics, we improve it by applying three new node ranking methods proposed in *5.1*, *5.2*, and *5.3*. Three modified RW algorithms are proposed respectively, in which the iterative computation of the relative resource quality of a node is based on (13), (14), or (15), along with the qualities of its neighbors.

6. Topology Decomposition

In Section 5 we use six topological characteristics to design three new algorithms and three modified algorithms based on our new node ranking methods. In this section we further explore the use of topological characteristics in the VNE problem from another aspect.

Just like the productivity of modern economy is based on the division of labor, division has been proved to play a vital role in solving many difficult problems in all kinds of disciplines. Some previous heuristic VNE algorithms have applied the theory of division in different ways. Most of the VNE algorithms can be divided into two stages, firstly the node mapping stage and secondly the link mapping stage. Some VNE algorithms (e.g., [6])

differentiate nodes as different types, while some others (e.g., [7, 8]) divide the VNs into many sub-graphs.

Qing et al. [12] prune the topology of a VN by using the K-core decomposition algorithm to divide it into a core network and multiple edge networks, and try to leverage the respective advantages of different VNE algorithms at the same time. Intuitively, a network core is a set of nodes that are highly and mutually interconnected. For a binary network, the K-core of a graph is the maximal connected sub-graph comprising nodes with degree at least k, and is derived by recursively deleting the nodes with degree less than k. K-core decomposition is an important tool for the visualization of complex networks [24, 25].

However, K-core decomposition is not adequate to the weighted networks and some specific topologies (e.g., star topology or tree topology), as a result of only considering the number of connections, i.e., the characteristic "degree". We demonstrate it in Fig. 3(a), where there is a VNR with 9 nodes. A thick line denotes a strong link with high bandwidth while a thin one denotes the opposite. As we can see, node *a*, *b*, *c*, *e*, and *h* all have a strong link to each other, which implies some structural or functional importance in the network. After applying the K-core decomposition to this topology, node *d*, *e*, *f*, *g*, *h*, and *i* are all peeled off as nodes of edge networks, and only node *a*, *b*, and *c* are included in the core network. Obviously this is not a satisfactory division where node *e* and node *h* are excluded from the core network.

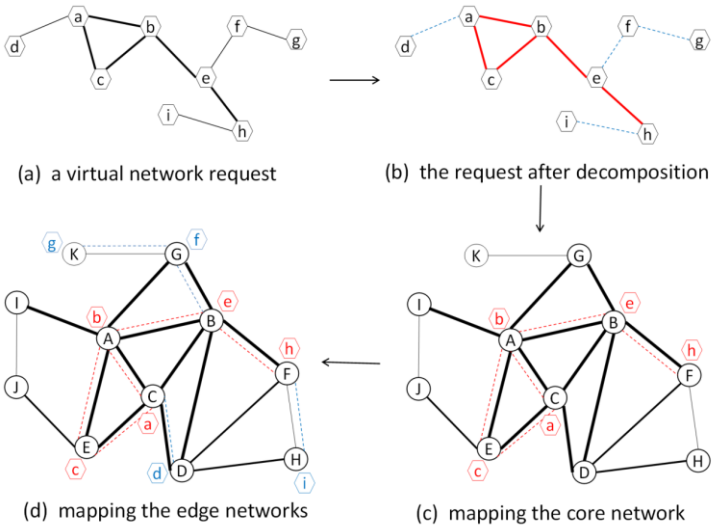


Fig. 3. An example of KS-core decomposition and mapping of a virtual network request

Here, we advocate utilizing multiple topological characteristics to devise new algorithms of topology decomposition, in order to better disentangle the hierarchical structure of a VN.

The strength, which is complementary to the degree, naturally becomes our first choice. A node with high degree has many connections while a node with high strength has strong connections. Therefore, the strength can be treated as a more sophisticated version of the degree, where a node does not depend on the number of links incident to it but on the qualities of those links. By taking into consideration both the number of connections and their qualities, we propose a new KS-core decomposition algorithm, in which the "S" stands for the "Strength".

We implement the KS-core decomposition by recursively removing the nodes whose degree equals to 1 and strength is less than the Strength Threshold, and pruning the corresponding links as well. The Strength Threshold is a parameter predefined according to

Algorithm 3 KS-core decomposition algorithm

```
1.  Take the virtual network request to be decomposed.
2.  for every node of the virtual network request do
3.      Calculate the degree and strength of the node.
4.      if (degree = 1) and (strength ≤ Strength Threshold) then
5.          Mark and prune the node and corresponding links.
6.          the sum of nodes --;
7.          if the sum of nodes ≤ Minimum then
8.              break;
9.          end if
10.     else if do not exist a node whose degree equals 1 and strength is less than the
        Strength Threshold then
11.         break;
12.     end if
13. end for
```

the topologies of VNRs and the substrate network (such as the average bandwidth of links, the sum of nodes, and the linking probability of nodes), so as to properly differentiate the core network and the edge networks. The decomposition process will stop once no further removal is possible or the number of residual nodes in the VNR reaches the Minimum value, which should also be predefined to achieve the relative balance of nodes in core network and in edge networks. The remaining KS-core contains only nodes with strength greater than the Strength Threshold and degree at least 2, and the pruned parts are multiple edge networks. We describe the KS-core decomposition algorithm in Algorithm 3.

In **Fig. 3(b)**, we show the result of the same VNR in **Fig. 3(a)** after applying the KS-core decomposition algorithm. Five nodes incident to strong links, represented by red solid lines, are included in the core network. The residual parts, represented by blue dashed lines, are three independent edge networks. **Fig. 3(c)** and **Fig. 3(d)** depict the mapping of the core network and edge networks respectively.

To compare KS-core decomposition algorithm with K-core decomposition algorithm in the evaluation, we employ the same node mapping algorithms and link mapping algorithms as that in [12]. The evaluation results will be presented in the next section.

7. Performance Evaluation

In this section, we first describe the performance evaluation environment that is configured similar to previous work [9, 11, 12], and then present our main evaluation results. Our evaluation focuses primarily on quantifying the benefits of our different algorithms.

7.1. Evaluation Environment

Because the actual substrate networks and VNs are not well understood yet, we use GT-ITM tool [26] to generate substrate networks and VNRs in our evaluation as most previous work did. Each substrate network is configured to have 100 nodes with over 500 links, which is about the scale of a medium-sized ISP. And the corresponding CPU and bandwidth resources are real numbers uniformly distributed from 50 to 100. The arrival of VNRs is modeled following a Poisson process with an average arrival rate of 5 VNs per 100 time units. We assume that each VN has an exponentially distributed duration with an average of 500 time units, while the delay is 200 time units. The number of nodes in a VNR is configured as a

TABLE 1. ALGORITHMS IN COMPARISON

Notation	Algorithm Description	Simulator
BL	The algorithm in [9] without employing path migration, as the 1st baseline algorithm	ND#
CL	The classical closeness centrality algorithm in [13], as the 2nd baseline algorithm	ND#
ND	The mapping algorithm with degree in Section *5.1*	ND#
NDC	The mapping algorithm with degree and closeness in Section *5.2*	ND#
NDCB	The mapping algorithm with degree, closeness, and betweenness in Section *5.3*	ND#
RW	The Random Walk algorithm in [11] as the 3rd baseline algorithm	RW#
RW-ND	Modified version of RW using the algorithm ND	RW#
RW-NDC	Modified version of RW using the algorithm NDC	RW#
RW-NDCB	Modified version of RW using the algorithm NDCB	RW#
K-core	Hybrid mapping algorithm with K-core decomposition in [12], as the 4th baseline algorithm	KS#
KS-core(*)	Hybrid mapping algorithm with KS-core decomposition in Section 6, predefined with different Strength Thresholds in parentheses	KS#

uniform distribution between 5 and 20. Pairs of virtual nodes are randomly connected by links with the probability of 0.5, which means a n-node VN has $n(n-1)/4$ links on average. CPU and bandwidth requirements of virtual nodes and virtual links are real number uniformly distributed between 1 and 50. Each experiment runs ten different instances, and each instance lasts for more than 56000 time units, corresponding to about 2800 VNRs. The arithmetic mean of ten instances is recorded as the final result.

We implement three VNE simulators to respectively compare our 7 proposed topology-aware algorithms with 4 corresponding existing algorithms. The notations we refer to different algorithms are enumerated in **Table 1**, along with the simulators used. The ND# simulator is a modified version of the VNE simulator [27] designed in [9], the RW# simulator is a modified version of the simulator in [11], and the KS# simulator is a modified version of the simulator in [12]. Performance metrics in comparison are three objectives presented in Section 3, i.e., the long-term average revenue, acceptance ratio, and RC ratio.

7.2. Evaluation Results

The evaluation consists of three parts with three simulators respectively: (1) using the ND# simulator to compare two baseline algorithms BL in [9] and CL in [13] with our three algorithms ND, NDC, and NDCB; (2) using the RW# simulator to compare the baseline algorithm RW in [11] with three modified algorithms RW-ND, RW-NDC, and RW-NDCB; (3) using the KS# simulator to compare the baseline algorithm K-core in [12] with KS-core decomposition algorithm. In all the following figures, the performance metrics are plotted against the time.

7.2.1. Mapping Algorithms with Multiple Characteristics

Fig. 4 shows the long-term average revenue, acceptance ratio, and RC ratio of two baseline algorithms BL, CL, and three topology-aware algorithms ND, NDC, and NDCB. All the curves reach a relatively steady state after about 10000 time units.

In the left part of **Fig. 4**, the average revenue of algorithm ND is remarkably higher than that of the baseline algorithm BL, and algorithm NDC further improves it. The baseline algorithm CL shows impressively high average revenue as well, while algorithm NDCB has

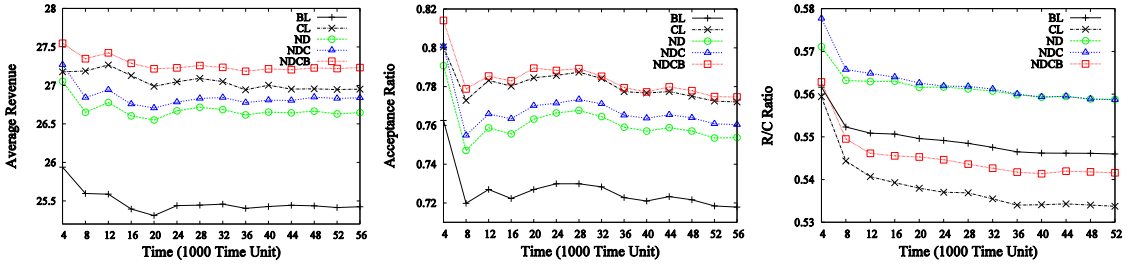


Fig. 4. average revenue, acceptance ratio, and revenue/cost ratio of 5 algorithms in comparison

the relatively highest average revenue, which is about 8% higher than BL. We can see that a more comprehensive algorithm that considers multiple topological characteristics does improve the long-term average revenue.

The middle part of **Fig. 4** shows the results of the long-term acceptance ratio, which reveal a similar relationship to these of the long-term average revenue, and also confirm that high acceptance ratio usually guarantees high revenue. However, there is an exception. The baseline algorithm CL has almost the same high acceptance ratio as NDCB, which is about 8% higher than BL. Compared with the average revenue in the left, CL accepts as many VNRs as NDCB does, but obtains relatively lower average revenue. It indicates that the use of closeness centrality can increase the acceptance ratio, but has a tendency of accepting relatively smaller VNRs in the long term. This is because CL only chooses the nodes closer to the center of topology, and does not consider the load balance of the substrate network, which may easily lead to the fragmentation of substrate network resources. Thus, CL can accept the VNRs with smaller sizes, but has to reject those VNRs with large sizes. NDCB accepts more VNRs and achieves higher average revenue than others at the same time.

The right part of **Fig. 4** shows the results of the long-term RC ratio, which demonstrate a little different relationship of 5 algorithms in comparison. The curves of RC ratios all have a slow downward trend because of the gradual fragmentation of the substrate network over time. As a result, virtual links of VNRs have to be embedded into longer physical paths. The RC ratios of ND and NDC are remarkably higher than that of BL. However, CL has impressively low RC ratio, and NDCB has relatively higher RC ratio that is still lower than BL. NDC is about 5% higher than CL. We can see that ND and NDC improve the efficiency of embedding, but CL and NDCB achieve higher average revenues and acceptance ratios by sacrificing the efficiency more or less.

To summarize, our algorithm NDCB has relatively highest long-term average revenue and acceptance ratio, but these gains come at a penalty in the long-term RC ratio. Our algorithm ND improves the long-term average revenue, acceptance ratio, and RC ratio at the same time, while NDC further improves them. The baseline algorithm CL performs well in the long-term average revenue and acceptance ratio, but has a rather poor RC ratio.

7.2.2. Modified Algorithms with Multiple Characteristics

Fig. 5 shows the long-term average revenue，acceptance ratio, and RC ratio of the baseline algorithm RW and three modified algorithms RW-ND, RW-NDC, and RW-NDCB.

The left part and middle part of **Fig. 5** reveal similar relationship of these algorithms with respect to the long-term average revenue and acceptance ratio. The RW-* algorithms all have higher average revenues and acceptance ratios than the original RW algorithm, while RW-NDC has similar average revenue and acceptance ratio as RW-ND. However, RW-NDCB does not achieve the highest average revenue and acceptance ratio as expected.

The iterative computation of obtaining the ranking of nodes in the RW and RW-* algorithms does not reveal the advantages of using certain topological characteristics, such as

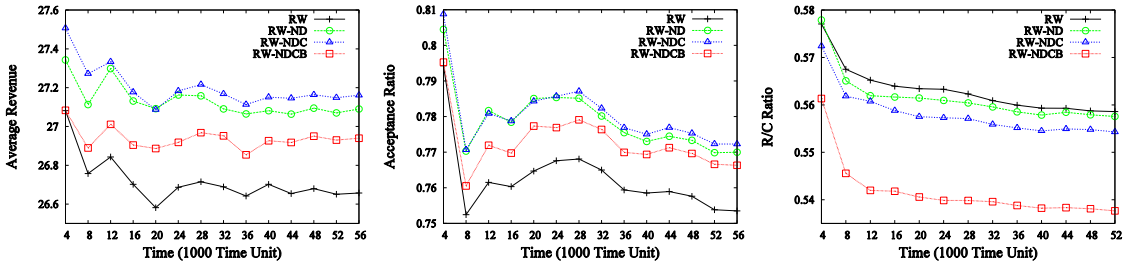


Fig. 5. average revenue, acceptance ratio, and revenue/cost ratio of 4 algorithms in comparison

betweenness centrality. Nevertheless, the use of multiple topological characteristics can still effectively improve the long-term average revenue and acceptance ratio of the RW algorithm.

The right part of **Fig. 5** shows the long-term RC ratio of 4 algorithms in comparison. The curves of RC ratio of RW-* algorithms, especially the algorithm RW-NDCB, are lower than that of the original RW algorithm. This indicates that RW-* algorithms sacrifice the efficiency more or less, and the use of degree, strength, and closeness centrality performs better with the RW algorithm than the use of betweenness centrality.

To summarize, the modified RW algorithms using our node ranking methods all achieve higher average revenue and acceptance ratio than the original RW algorithm, but these gains come at a penalty in the long-term RC ratio.

7.2.3. KS-core Decomposition Algorithm

Fig. 6 shows the long-term average revenues, acceptance ratio, and RC ratio of the K-core decomposition algorithm in [12] and our KS-core decomposition algorithm predefined with different Strength Thresholds.

The KS-core decomposition algorithm has a very slight effect on the acceptance ratio compared to the K-core decomposition algorithm, but an obvious effect on the average revenue and especially the RC ratio. When the Strength Threshold is set to be 60 or higher, the KS-core decomposition algorithm has relatively the best performance in terms of the long-term average revenue and RC ratio (about 7% improvement). The performance cannot be further improved with higher Strength Threshold, because the number of edge nodes with their strengths below the Strength Threshold will not increase any more. When the Strength Threshold is 50, the KS-core decomposition algorithm also has a relatively better performance in terms of the long-term average revenue and RC ratio than the K-core decomposition algorithm. However, when the Strength Threshold is 40, the long-term average revenue is hardly improved, and the RC ratio is lower than the K-core decomposition algorithm. This effect is more distinct when the Strength Threshold is 30. To achieve the best performance of our KS-core decomposition algorithm, the Strength Threshold should be properly configured according to the topologies of VNRs and the substrate network.

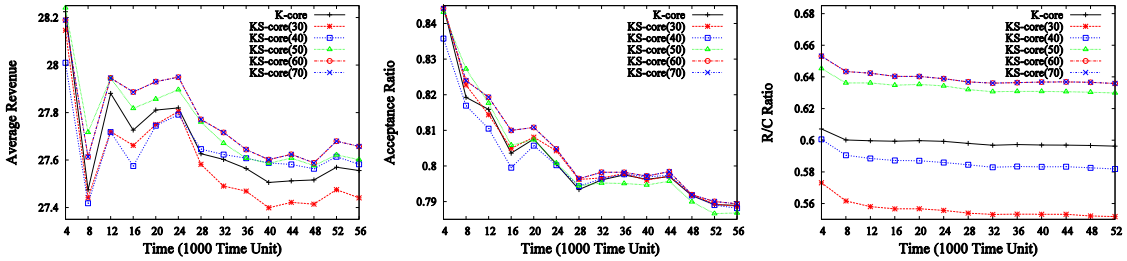


Fig. 6. average revenue, acceptance ratio, and revenue/cost ratio of 2 Algorithms in comparison

8. Conclusion

The VNE problem is an open issue in network virtualization. We introduce six topological characteristics complementary to each other, which measure the relative influence or importance of nodes in the substrate network and VNs from different aspects. Based on the introduced topological characteristics, we propose three topology-aware VNE algorithms and three corresponding modified algorithms for the RW algorithm, by leveraging the respective advantages of different characteristics. On the other hand, KS-core decomposition algorithm is devised to utilize the topological characteristics from the aspect of topology decomposition in the VNE problem. Extensive simulations between our 7 proposed algorithms and 4 baseline algorithms demonstrate that our algorithms better coordinate node and link mapping, and substantially increase the long-term average revenue, acceptance ratio, and RC ratio compared to the previous algorithms.

In our future work, the six characteristics proposed should be further analyzed in order to understand their relationships and their exact influence over the practical VNE algorithms, and avoid mutual interferences. Furthermore, topology decomposition with multiple topological characteristics will be further studied.

References

[1] N. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol.54, no. 5, pp. 862–876, 2010. Article(CrossRef Link)

[2] T. Anderson, L. Peterson, S. Shenker and J. Turner, "Overcoming the Internet impasse through virtualization," *IEEE Computer Magazine*, vol. 38, no. 4, pp.34–41, 2005. Article(CrossRef Link)

[3] J. Turner and D. Taylor, "Diversifying the Internet," in *Proc. of IEEE GLOBECOM*, vol. 2, pp. 755–760, 2005. Article(CrossRef Link)

[4] N. Feamster, L. Gao and J. Rexford, "How to lease the Internet in your spare time," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 1, pp. 61-64, 2007. Article(CrossRef Link)

[5] A. Bavier, N. Feamster, M. Huang, L. Peterson and J. Rexford, "In VINI veritas: Realistic and controlled network experimentation," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 3-14, 2006. Article(CrossRef Link)

[6] W. Szeto, Y. Iraqi and R. Boutaba, "A multi-commodity flow based approach to virtual network resource allocation," in *Proc. of IEEE GLOBECOM*, vol. 6, pp. 3004-3008, 2003. Article(CrossRef Link)

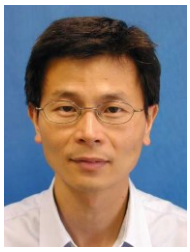
[7] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proc. of IEEE INFOCOM*, 2006. Article(CrossRef Link)

[8] I. Houidi, W. Louati and D. Zeghlache, "A distributed virtual network mapping algorithm," in *Proc. of IEEE ICC*, pp. 5634-5640, 2008. Article(CrossRef Link)

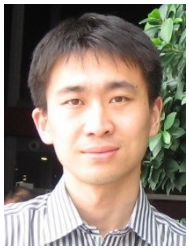
[9] M. Yu, Y. Yi, J. Rexford and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 17-29, 2008. Article(CrossRef Link)
[10] N. Chowdhury, M. Rahman and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. of IEEE INFOCOM*, pp. 783–791, 2009. Article(CrossRef Link)
[11] X. Cheng, S. Su, F. Yang et al., "Virtual network embedding through topology-Aware node ranking," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 38-47, 2011. Article(CrossRef Link)
[12] S. Qing, J. Liao, J. Wang, X. Zhu and Q. Qi, "Hybrid virtual network embedding with K-core decomposition and time-oriented priority," in *Proc. of IEEE ICC*, pp. 2695-2699, 2012. Article(CrossRef Link)
[13] Z. Wang, Y. Han, T. Lin, H. Tang and S. Ci, "Virtual network embedding by exploiting topological information," in *Proc. of IEEE GLOBECOM*, 2012. Article(CrossRef Link)
[14] P. Marchetta, P. Mérindol, B. Donnet, A. Pescapé and J. Pansiot, "Topology discovery at the router level: a new hybrid tool targeting ISP networks," *IEEE Journal on Selected Areas in Communications*, vol. 29, no.9, pp.1776-1787, 2011. Article(CrossRef Link)
[15] D. Stezenbach, M. Hartmann and K. Tutschku, "Parameters and challenges for virtual network embedding in the future internet," in *Proc. of IEEE NOMS*, pp. 1272-1278, 2012. Article(CrossRef Link)
[16] M. Newman, *Networks: An Introduction*, Oxford University Press, Oxford, UK, 2010. Article(CrossRef Link)
[17] T. Opsahl, F. Agneessens and J. Skvoretz, "Node centrality in weighted networks: generalizing degree and shortest paths," *Social Networks*, vol. 32, no. 3, pp. 245-251, 2010. Article(CrossRef Link)
[18] L. Freeman, *The development of social network analysis*, Empirical Press, Vancouver, Canada, 2004.
[19] P. Hagmann, L. Cammoun, X. Gigandet, et al., "Mapping the structural core of human cerebral cortex," *PLoS biology*, vol. 6, no. 7: e159, 2008. Article(CrossRef Link)
[20] D. Eppstein, "Finding the k shortest paths," *SIAM Journal on computing*, vol. 28, no. 2, pp. 652-673, 1998. Article(CrossRef Link)
[21] R. K. Ahuja, T. L. Magnanti, J. B. Orlin and K. Weihe, *Network flows: theory, algorithms, and applications*, Prentice Hall, 1993.
[22] J. Castro and N. Nabona, "An implementation of linear and nonlinear multicommodity network flows," *European Journal of Operational Research*, vol. 92, no.1, pp.37-53, 1996. Article(CrossRef Link)
[23] D. Austin, "How Google finds your needle in the web's haystack," *American Mathematical Society Feature Column*, pp.10-12, 2006.
[24] S. N. Dorogovtsev, A. V. Goltsev and J. F. F. Mendes, "K-core organization of complex networks," *Physical review letters*, vol. 96, no. 4, 2006. Article(CrossRef Link)
[25] J. I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat and A. Vespignani, "Large scale networks fingerprinting and visualization using the k-core decomposition," *Advances in neural information processing systems*, pp. 41–50, 2005.
[26] E. Zegura, K. Calvert and S. Bhattacharjee, "How to model an Internetwork," in *Proc. of IEEE INFOCOM*, 1996. Article(CrossRef Link)
[27] Virtual Network Embedding Simulator. https://github.com/minlanyu/embed

Jianxin Liao obtained his PhD degree from University of Electronics Science and Technology of China in 1996. He is currently the dean of Network Intelligence Research Center and the full professor of State Key laboratory of Networking and Switching Technology in Beijing University of Posts and Telecommunications. He has published hundreds of research papers and several books, and has been granted dozens of patents for inventions. His prizes include the Premier's Award of Distinguished Young Scientists from National Natural Science Foundation of China in 2005, and the Specially-invited Professor of the "Yangtse River Scholar Award Program" by Ministry of Education of China in 2009. His main creative contributions include mobile intelligent network, service network intelligent, networking architectures and protocols, and multimedia communication. These achievements were conferred the "National Prize for Progress in Science and Technology" twice, respectively in 2004 and 2009.

Min Feng is a PhD candidate in Communication and Information System, from the State Key Lab of Networking and Switching Technology in Beijing University of Posts and Telecommunications. His research interests include network virtualization, cloud computing, and data center network.



Tonghong Li obtained his PhD degree from Beijing University of Posts and Telecommunications in 1999. He is currently an assistant professor with the department of computer science, Technical University of Madrid, Spain. His main research interests include resource management, distributed system, middleware, wireless networks, and sensor networks.



Jingyu Wang obtained his PhD degree from Beijing University of Posts and Telecommunications in 2008. He is currently an associate professor in Beijing University of Posts and Telecommunications. His research interests span broad aspects of performance evaluation for Internet and overlay network, consumer electronic, traffic engineering, image/video coding, and multimedia communication over wireless network.

Sude Qing received his PhD degree in Computer Science from Beijing University of Posts and Telecommunications in 2013. His research interests include cloud computing and network virtualization.