Efficient Virtual Network Embedding Via Exploring Periodic Resource Demands

Zichuan Xu Weifa Liang Qiufen Xia
Research School of Computer Science
Australian National University, Canberra, ACT 0200, Australia
Email: edward.xu@anu.edu.au, wliang@cs.anu.edu.au, qiufen.xia@anu.edu.au

***Abstract*—Cloud computing built on virtualization technologies promises provisioning elastic computing and communication resources to enterprise users. To share cloud resources efficiently, embedding virtual networks of different users to a distributed cloud consisting of multiple data centers (a substrate network) poses great challenges. Motivated by the fact that most enterprise virtual networks usually operate on long-term basics and have the characteristics of periodic resource demands, in this paper we study the virtual network embedding problem by embedding as many virtual networks as possible to a substrate network such that the revenue of the service provider of the substrate network is maximized, while meeting various Service Level Agreements (SLAs) between enterprise users and the cloud service provider. For this problem, we propose an efficient embedding algorithm by exploring periodic resource demands of virtual networks, and employing a novel embedding metric that models the workloads on both substrate nodes and communication links if the periodic resource demands of virtual networks are given; otherwise, we propose a prediction model to predict the periodic resource demands of these virtual networks based on their historic resource demands. We also evaluate the performance of the proposed algorithms by experimental simulation. Experimental results demonstrate that the proposed algorithms outperform existing algorithms, improving the revenue from 10% to 31%.**

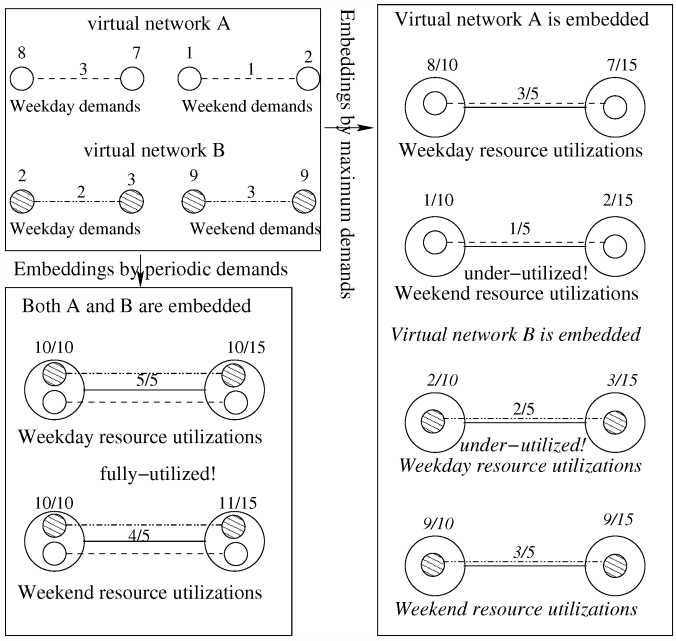


Fig. 1. A motivated example

I. Introduction

Enterprises nowadays are embracing a new computing paradigm by outsourcing their IT service networks as virtual networks to clouds for cost savings. For example, a company operating video conferencing services could run on a virtual network with a stringent quality of service (QoS) requirement by allocating dedicated resources and employing robust routing protocols, whereas a university delivering online courses for distance education may run a virtual network with guaranteed bandwidth for real-time delivery of its online courses. Both of such virtual networks can be run on a substrate network that consists of multiple data centers. A fundamental problem related to these applications is to devise an efficient algorithm to accommodate as many virtual networks as possible in the substrate network such that the operational cost of the cloud service provider is minimized. We refer to this problem as the *Virtual Network Embedding* (VNE) problem, which has been extensively studied in the past couple of years [2], [7], [10].

Most existing studies of the VNE problem in literature focused on resource provisions by reserving the maximum resource demands for each virtual network throughout its whole lifetime [6], [8], [22], [27], [25]. Such a resource provision scheme however causes up to 85 percent of cloud resources under-utilized in most time, resulting in enormous resource wastage and economic loss [21]. Fortunately, nearly 90% of enterprise IT services exhibit periodic resource demand patterns [11]. By making use of this property, the resource utilization ratio in the cloud can be substantially improved if the demanded resources by different virtual networks can shared. We illustrate this observation by an example as shown in Fig. 1, where a virtual network A providing office users with virtual desktop services usually experiences low-workloads at weekends, whereas another virtual network B hosting online gaming services has high-workload at weekends due to high user demands. If embedding A and B by their maximum demands, only one of them can be embedded into the substrate network. However, they can be serviced if their time-varying resource demands are complementary. Due to the heterogeneity of substrate resources and the unknownness of periodic resource demands of virtual networks, it poses a great challenge to embed as many virtual networks as possible to a substrate network to maximize the cloud resource utilization ratio.

Despite that there are existing works dealing with virtual net-

978-1-4799-3780-6/14/$31.00 ©2014 IEEE

work embedding, to the best of our knowledge, we are the first to explore the periodic resource demands of virtual networks and make use of this property to allocate distributed cloud resources among virtual networks. In addition, we propose a novel embedding metric that models the dynamic workloads of both substrate nodes and links in a substrate network.

The main contributions of this paper are as follows. We first propose an embedding algorithm for the VNE problem by employing a novel embedding metric that models dynamic workloads of cloud resource usages, assuming that the periodic resource demands of each virtual network are given. We then devise an embedding algorithm without the knowledge of periodic resource demands through resource demand prediction. We finally evaluate the performance of the proposed algorithms by experimental simulations. Experimental results show that the proposed algorithms outperform existing algorithms, improving the revenue of the cloud service provider from 10% to 31%.

The remainder of the paper is organized as follows. Section II introduces related work, followed by the system model and problem definitions in Section III. Sections IV and V propose VNE algorithms with and without the periodic resource demands of virtual networks. Section VI evaluates the performance of the proposed algorithms through experimental simulations. The conclusion is given in Section VII.

II. Related Work

Most existing solutions to the VNE problem can be classified into two categories: *static* and *dynamic* resource provisioning. Static resource provisioning assumes that the resource demands of each virtual network do not change during the lifetime of the virtual network, whereas dynamic resource provisioning deals with the embedding of virtual networks with dynamic resource demands, topologies, and sizes. Most existing studies in literature focused on static resource provisions [4], [6], [8], [9], [18], [22], [25], [27]. For example, Zhu *et. al.* [27] proposed a VNE algorithm for workload balancing by introducing a node/link-stress concept, and jointly considered the workloads on each node and the link incident to it. Chowdhury *et. al.* [8] devised a coordinated node and link mapping by reducing the problem to a multi-commodity flow problem under the constraint that each virtual node has only several candidate geographical locations. Cheng *et. al.* [6] proposed an embedding algorithm, by using a similar idea to the one in Google's PageRank algorithm, where both substrate and virtual nodes are ranked according to their available resources and the quality of link connections. Lischka *et. al.* [18] devised an online VNE algorithm by utilizing the subgraph isomorphism detection with the aim of maximizing the revenue-to-cost ratio, where the revenue is the total amount of virtual resources requested by virtual networks and the cost is the total amount of substrate resources spent in accommodating the virtual networks. Other static approaches take different perspectives on the VNE problem, e.g., splittable path routing [22], embedding one virtual node onto several substrate nodes [25], embedding across different substrate networks [14], distributed and automatic embedding [15], or avoiding resource fragmentation in the substrate network [9].

There are also several studies focusing on dynamic resource provisioning by reallocating under-utilized resources to other virtual network requests [5], [20], [23], [24], [26]. Zhang *et. al.* [23] re-examined the VNE problem by considering opportunistic resource sharing and topology-aware node ranking. They assumed each virtual network has basic and maximum demands with certain probabilities, Such an assumption is not realistic as it is unlikely that users can provide detailed resource demands information in advance. The other dynamic resource provisioning approaches however perform periodic reconfigurations/migrations of implemented virtual networks, which may not be feasible due to incurred high migration costs or the violations of the agreed SLA requirements [3]. For example, Houidi *et. al.* [16] proposed an adaptive VNE algorithm that dynamically identifies new candidate substrate resources to cater dynamic topologies and dynamic communication requirements of virtual networks. Similarly, approaches in [5], [26] dealt with evolving virtual networks in terms of topologies and resource demands through redeployments of embedded virtual networks. Sun *et. al.* [20] devised virtual network migration algorithms to deal with evolving virtual networks. Zhang *et. al.* [24] studied a scenario that both the demands of virtual networks and the capacity of a substrate network will change over time, which however is not realistic either since the capacity of a data center usually does not change over time. Unlike previous works on static resource provisions, in this paper we deal with dynamic resource provisions for virtual networks, by exploring periodic resource demands. The essential difference between our work and existing works lies in a novel embedding metric that can model the workloads of both substrate nodes and substrate links accurately over time.

III. Preliminaries

A. Substrate and virtual networks

A *substrate network* is represented by a node-and-edge weighted undirected graph $G^s = (N^s, E^s)$, where N^s and E^s are the sets of substrate nodes and links, respectively. Denote by n^s a substrate node in N^s and e^s a link in E^s. Each n^s represents a data center and each e^s denotes a communication link between the two data centers corresponding to its two endpoints. Denote by $C(n^s)$ the capacity of computing resource in n^s and $B(e^s)$ the bandwidth capacity on e^s.

A *virtual network* can be represented by a node-and-edge weighted undirected graph $G^v = (N^v, E^v)$, where N^v and E^v are the sets of virtual nodes and virtual links. Each virtual node $n^v \in N^v$ represents a set of virtual machines that host specific applications. Each virtual edge $e^v \in E^v$ represents a communication link between two virtual nodes. Denote by $C(n^v)$ and $B(e^v)$ the *maximum amounts of computing and communication resource demands* by virtual node n^v and virtual link e^v, respectively.

Assume that time is divided into equal *time intervals*. Each time interval is further divided into equal numbers of *time slots*. Let i be the current time interval and T the number of time slots in each interval. Assume that virtual network requests

from users arrive in the system one by one, but they are only processed in the beginning of the next time slot after their arrival. Given a virtual network G^v with a duration $\tau(G^v)$ in the granularity of weeks or months, it is embedded to the substrate network prior to the expiration of its specified duration. Fig. 2 gives an example of virtual network embedding.

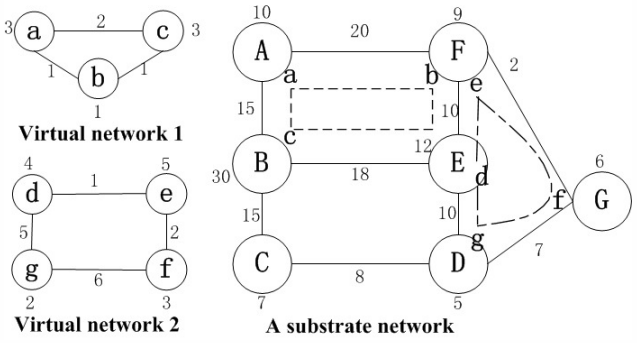


Fig. 2. Virtual network embedding

B. Periodic resource demands

Most enterprise IT services exhibit periodic demand patterns [11]. For instance, an enterprise that provides email services for a university has weekly resource demand patterns due to the weekly activity patterns of university users. Although periodic resource demands of each virtual network typically are not known when the virtual network request arrives, they can be predicted by analyzing its resource demand history, using offline profiling and online calibration [19]. Denote by $\hat{C}(n^v, i, t)$ and $\hat{B}(e^v, i, t)$ the predicted computing and bandwidth resource demands of a virtual node $n^v \in N^v$ and a virtual link $e^v \in E^v$ at the tth time slot in interval i. Let $C(n^v, i, t)$ be the actual amount of demanded computing resources of n^v at time slot t of interval i, which is no greater than its maximum resource demand $C(n^v)$. Similarly, the network bandwidth demand of virtual link e^v at time slot t of interval i is represented by $B(e^v, i, t)$ which is no greater than $B(e^v)$, for all t with $1 \leq t \leq T$. The amounts of available resources of substrate network G^s can be derived from the accumulative resources allocated to all embedded virtual networks in it at time slot t of interval i. Denote by $P(n^s, i, t)$ and $P(e^s, i, t)$ the amounts of available computing and bandwidth resources in node n^s and link e^s at time slot t of interval i.

C. Revenue and cost models

The revenue of a cloud service provider received by embedding virtual networks can be defined differently according to different economic models. Similar to the revenue models in previous studies [8], [27], [23], the revenue model adopted in this paper by embedding virtual network G^v in a time interval is defined as the sum of amounts of computing and bandwidth resources it requested. Denote by $\mathbb{R}(G^v, i)$ the revenue of admitting virtual network G^v at time interval i, then

$$\mathbb{R}(G^v, i) = \sum_{n^v \in N^v} C(n^v) + \sum_{e^v \in E^v} B(e^v). \tag{1}$$

To provide the demanded computing and bandwidth resources to a virtual network G^v, the cloud service provider consumes its resources such as electricity, software and hardware that incur its service costs. Thus, the cost of embedding a virtual network G^v is the sum of amounts of resources allocated to the virtual network per time interval. Denote by $\mathbb{C}(G^v, i)$ the cost of an embedded virtual network G^v in interval i, then

$$\mathbb{C}(G^v, i) = \sum_{t=1}^{T} \Big(\sum_{n^v \in N^v} C(n^v, i, t) + \sum_{e^v \in E^v} \sum_{e^s \in E^s} l_{e^s}^{e^v} B(e^v, i, t) \Big), \tag{2}$$

where $l_{e^s}^{e^v}$ is 1 if virtual link $e^v \in E^v$ is embedded to a path in G^s and $e^s \in E^s$ is a link in the path, and 0 otherwise.

An efficient embedding of the cloud service provider is to maximize its revenue while keeping its service cost minimized. We thus define the revenue-to-cost ratio $\eta(G^v, i)$ to quantify the efficiency of embedding of virtual network G^v at time interval i as follows.

$$\eta(G^v, i) = \frac{\mathbb{R}(G^v, i)}{\mathbb{C}(G^v, i)}. \tag{3}$$

Following the similar definitions given in [22], the *accumulated revenue, cost, and revenue-to-cost ratio* of virtual network G^v are defined by $\sum_{i'=1}^{\tau(G^v)} \mathbb{R}(G^v, i')$, $\sum_{i'=1}^{\tau(G^v)} \mathbb{C}(G^v, i')$ and $\sum_{i'=1}^{\tau(G^v)} \eta(G^v, i')$, respectively.

D. Problem definitions

Given an interval i consisting of T time slots, assume that virtual network requests arrive one by one without the knowledge of future arrivals. Let $\mathcal{G}(i, t)$ be the candidate set of virtual network requests to be embedded to $G^s(N^s, E^s)$ at the tth time slot of interval i. For each virtual network $G^v \in \mathcal{G}(i, t)$ with the given node and link resource demands, $C(n^v, i', t')$ and $B(e^v, i', t')$, and its duration $\tau(G^v)$, for all i' and t' with $i \leq i' < i + \tau(G^v)$ and $1 \leq t' \leq T$, the *virtual network embedding problem with the knowledge of periodic resource demands* is to embed as many virtual networks in $\mathcal{G}(i, t)$ as possible to the substrate network G^s such that the revenue of the cloud service provider of G^s is maximized while keeping its service cost minimized, subject to meeting the resource demands of each virtual network at each time slot. Similarly, the *virtual network embedding problem without the knowledge of periodic resource demands* is to embed as many virtual networks in $\mathcal{G}(i, t)$ as possible to the substrate network without the knowledge of periodic resource demands, such that the revenue is maximized while the cost of all implemented virtual networks is minimized, subject to the constraint that the *resource violation ratio* of each virtual network G^v is controlled within a given threshold $\sigma(G^v)$, where the *resource violation ratio* of G^v is the amount of violated resource to the amount of total resource demands of G^v throughout its lifetime. For example, given a virtual network demanding one unit of resources at each time slot of its 10-time-slot lifetime, its resource violation ratio will be 10%, if it is provided with 0.5 unit resource for two time slots and one unit for the rest.

IV. Algorithm with periodic resource demands

In this section we consider the embedding of virtual networks with periodic resource demands. We first embed a virtual network $G^v(N^v, E^v)$ to the substrate network $G^s(N^s, E^s)$. We then devise an algorithm for embedding multiple virtual networks in $\mathcal{G}(i,t)$ at each time slot t to G^s.

A. Embedding a virtual network

Given a virtual network G^v, an embedding metric is needed to evaluate the current workload of G^s. Such a metric considers not only the amount of available resources but also the utilization ratio of the resources of G^s. In the following, we first define a metric to model the workloads of substrate nodes and links. We then devise an algorithm to embed a virtual network with static resource demands based on the designed embedding metric. We finally extend the embedding algorithm for virtual networks with periodic resource demands.

We start by proposing an embedding metric to evaluate the current workload of G^s. The *embedding ability* of a substrate node n^s in admitting a virtual node is jointly determined by the amount of available computing resources and the utilization ratio of the computing resources of n^s. *The marginal gain* of the embedding ability of n^s is diminishing with the increase of its utilization ratio, since the larger the proportion of its resources is occupied, the higher the risk of SLA violations the substrate node n^s faces. The embedding ability of a substrate link can be defined similarly. We here use an exponential function to model the embedding ability of a virtual node or a virtual link. Recall that $P(n^s, i, t)$ is the amount of available computing resources at substrate node n^s. Denote by $\Phi(n^s)$ the embedding metric of substrate node n^s, then

$$\Phi(n^s) = P(n^s, i, t) \cdot a^{\frac{P(n^s,i,t)}{C(n^s)}}, \tag{4}$$

where $a > 1$ is a constant, and $\frac{P(n^s,i,t)}{C(n^s)}$ is a complementary ratio to the utilization ratio of n^s. This exponential metric favors allocating each virtual node to one substrate node with a large amount of available resources and a low utilization ratio, while the metric $\Phi(e^s)$ of substrate link $e^s \in E^s$ is defined similarly,

$$\Phi(e^s) = P(e^s, i, t) \cdot b^{\frac{P(e^s,i,t)}{B(e^s)}}, \tag{5}$$

where $b > 1$ is a constant and $P(e^s, i, t)$ is the amount of available bandwidth of e^s. Notice that a virtual link e^v may be mapped to a path p in G^s consisting of several substrate links. To evaluate the (bandwidth) workload of p, we define the 'length' of p as the sum of the lengths of substrate links in p. Let $d(e^s)$ be the length of each link e^s, then,

$$d(e^s) = \begin{cases} \frac{1}{\Phi(e^s)} & \text{if } \Phi(e^s) > 0, \\ \infty & \text{if } \Phi(e^s) = 0. \end{cases} \tag{6}$$

This implies a shorter substrate link has more available bandwidth; otherwise, the link is discouraged to be used. The length of path p thus is defined as

$$d(p) = \sum_{e^s \in p} d(e^s). \tag{7}$$

We continue by devising an algorithm to embed a virtual network with static resource demands based on the designed embedding metric. Most embedding algorithms for embedding a virtual network $G^v(N^v, E^v)$ with static resource demands to $G^s(N^s, E^s)$ are as follows. Embed each virtual node in V^v to a different substrate node N^s and each virtual link in E^v to a path in G^s by different embedding metrics [8], [23], [27]. As this problem is notoriously NP-hard [10], following most existing studies, our embedding algorithm proceeds similarly by embedding virtual nodes in N^v first, followed by embedding virtual links in E^v.

We first embed virtual nodes in N^v. To this end, we select a cluster of substrate nodes for virtual nodes in N^v and embed each virtual node into a different substrate node in the cluster. The cluster is constructed dynamically. In such a cluster, each selected substrate node should have not only great embedding ability but also a shorter path length to other substrate nodes in the cluster, since the shorter the length between two selected substrate nodes, the less the bandwidth resource will be consumed on the path. Therefore, we first find a *cluster center* with the greatest embedding ability in G^s to embed a virtual node with the maximum resource demand in G^v, and then find other substrate nodes (cluster members) one by one iteratively with a shorter length to the selected substrate nodes, by adopting a similar strategy in [27]. Specifically, to find the cluster center to which the first virtual node in G^v to be mapped, each substrate node $n^s \in N^s$ is assigned a rank which is jointly determined by its own embedding ability and the accumulative embedding abilities of its incident links. The rationale is that the virtual links incident to the virtual node will be embedded into the paths including the substrate links incident to the mapped substrate node. Let $L(n^s)$ be the set of substrate links incident to a substrate node $n^s \in N^s$. *The rank* of n^s, $NR(n^s)$, is defined as *the product of* embedding abilities of n^s, $\Phi(n^s)$, and the embedding abilities of links in $L(n^s)$. Then,

$$NR(n^s) = \Phi(n^s) \cdot \sum_{e^s \in L(n^s)} \Phi(e^s). \tag{8}$$

Similarly, we assign each virtual node $n^v \in N^v$ a rank $NR(n^v)$ which is defined as the product of its computing resource demand and the accumulative bandwidth resource demands of its incident virtual links, i.e.,

$$NR(n^v) = C(n^v) \cdot \sum_{e^v \in L(n^v)} B(e^v). \tag{9}$$

Let n_c^s and n_c^v be the chosen cluster center and virtual node by Eqs. (8) and (9). n_c^v then is embedded into n_c^s.

Having embedded n_c^v to n_c^s, we then find the other $|N^v| - 1$ cluster members iteratively. During each iteration, one of unembedded virtual nodes in N^v is embedded to a substrate node that has high embedding ability and a shortest length to those selected substrate nodes in the cluster. To this end, those not yet selected substrate nodes are ranked by the products of the inverse of $\Phi(n^s)$ and the accumulative length from substrate

node n^s to all the substrate nodes selected. Denote by $\kappa(n^s)$ the rank of n^s, then,

$$\kappa(n^s) = \frac{1}{\Phi(n^s)} \cdot \sum_{m^s \in N^s_{sel}} d(p_{n^s,m^s}), \tag{10}$$

where N^s_{sel} is the set of selected substrate nodes in the cluster, and p_{n^s,m^s} is the shortest path between substrate node n^s and a substrate node $m^s \in N^s_{sel}$. The rank of an yet-to-be embedded virtual node n^v in N^v can be defined as follows.

$$\kappa(n^v) = \frac{1}{C(n^v)} \cdot \sum_{m^v \in N^v_{emd}} \sum_{e^v \in p_{n^v,m^v}} d(e^v), \tag{11}$$

where N^v_{emd} denotes the set of virtual nodes that have been embedded, p_{n^v,m^v} is the shortest path between a virtual node n^v and a virtual node $m^v \in N^v_{emd}$, and $d(e^v)$, the length of virtual edge e^v, is defined by $\frac{1}{B(e^v)}$. A virtual node n^v with the lowest value of $\kappa(n^v)$ will be chosen and embedded to a substrate node with the lowest value of $\kappa(n^s)$. If there is a virtual node that has not been embedded after considering all substrate nodes, G^v will be rejected. The embedding of virtual links in E^v can be dealt similarly. Let p be a shortest path between two substrate nodes for virtual link e^v. If p does not have enough bandwidth to meet the bandwidth demand of e^v, the next shortest path is searched until no such a path is found and the request is rejected. The detailed procedure is shown in **Algorithm 1**.

Algorithm 1: *Embedding a virtual network with static resource demands*

Input: Given a virtual network $G^v(N^v, E^v)$, a substrate network $G^s(N^s, E^s)$, the computing resource demand $C(n^v)$ of each virtual node $n^v \in N^v$, and the bandwidth resource demand $B(e^v)$ of each virtual link $e^v \in E^v$.
Output: Embedding G^v or reject it
// Stage one: embed virtual nodes
Find the cluster center n^s_c with the maximum ranking in substrate network G^s by Eq. (8);
Find the virtual node n^v_c with the maximum resource demand in virtual network G^v by Eq. (9);
Embed virtual node n^v_c to the cluster center n^s_c;
$N^v_{emd} \leftarrow \{n^v_c\}$; /* the set of embedded virtual nodes*/
$N^s_{sel} \leftarrow \{n^s_c\}$; /* the set of selected substrate nodes*/
Embed the virtual nodes in set $N^v - N^v_{emd}$ to G^s iteratively, by embedding the virtual node with minimum $\kappa(n^v)$ to the substrate node with minimum $\kappa(n^s)$ during each iteration;
If all the substrate nodes in N^s are explored and there are still non-embedded virtual nodes in E^v, reject G^v and exit;
// Stage two: embed virtual links
for *each virtual link* $e^v \in E^v$ **do**
    Update the weight of substrate link $e^s \in E^s$ by Eq. (6);
    Let n^s_1 and n^s_2 be the substrate nodes that embed the two virtual nodes connected by e^v;
    Find a shortest path p from node n^s_1 to n^s_2;
    If p cannot satisfy the resource demand $B(e^v)$ of virtual link e^v, find next shortest path from n^s_1 to n^s_2;
    If no path can satisfy the demand of e^v, reject virtual network G^v, and exit;
end
Admit virtual network request G^v;

We now propose an algorithm that embeds a virtual network $G^v(N^v, E^v)$ with periodic resource demands to G^s in an interval i as follows. We construct $T+1$ graphs with different resource demands at different time slots, and pre-embed each constructed graph by **Algorithm 1**, where each time interval consists of T equal time slots. G^v is then embedded into G^s according to one of the $T+1$ different embeddings that leads to the maximum revenue-to-cost ratio. Specifically, we first construct T graphs with each having the resource demands of G^v at time slot t' in interval i' with $1 \le t' \le T$. Let $G^v(t') = (N^v_{t'}, E^v_{t'})$ be the node-and-edge weighted graph with resource demands of G^v at time slot t', where $N^v_{t'} = N^v$, $E^v_{t'} = E^v$, $C(n^v) = C(n^v, i', t')$ for each $n^v \in N^v_{t'}$ and $B(e^v) = B(e^v, i', t')$ for each $e^v \in E^v_{t'}$, where $1 \le t' \le T$ and $i \le i' \le i + \tau(G^v)$. Similarly, we use graph $G^v_m = (N^v_m, E^v_m)$ with $N^v_m = N^v$ and $E^v_m = E^v$ to denote virtual network G^v with average resource demands within an interval, where $C(n^v) = \frac{1}{T}\sum_{t'=1}^{T} C(n^v, i', t')$ for each $n^v \in N^v_m$ and $B(e^v) = \frac{1}{T}\sum_{t'=1}^{T} B(e^v, i', t')$ for each $e^v \in E^v_m$. Let $\mathcal{G}_{pre}$ be the set of above constructed $T+1$ graphs, i.e., $\mathcal{G}_{pre} = \{G^v(t') \mid 1 \le t' \le T\} \cup \{G^v_m\}$. Note that an embedding of a graph in $\mathcal{G}_{pre}$ is a feasible embedding only when it can be embedded into G^s and satisfies the resource demands of G^v at each time slot of an interval. Let G^v_{max} be the graph achieving the maximum revenue-to-cost ratio. Then, virtual network G^v is embedded to the substrate network according to the resource demands of G^v_{max}. Detailed algorithm is given by **Algorithm 2**.

Algorithm 2: *Embedding a virtual network with given periodic resource demands*

Input: Given a virtual network $G^v(N^v, E^v)$ at tth time slot of interval i, the substrate network G^s, and the periodic resource demands of G^v
Output: Embedding G^v or reject it
Construct $T+1$ graphs for G^v, where each of the first T graphs represents the resource demand of G^v at a time slot of an interval, and the last graph denotes the average resource demand of an interval;
Let $\mathcal{G}_{pre}$ be the set of all the $T+1$ graphs, i.e., $\mathcal{G}_{pre} \leftarrow \{G^v(t') \mid 1 \le t' \le T\} \cup \{G^v_m\}$;
Pre-embed each graph in $\mathcal{G}_{pre}$ by **Algorithm 1**;
if Algorithm 1 *rejects all graphs in* $\mathcal{G}_{pre}$ **then**
    Reject G^v and exit;
end
Identify an embedding in Step 3 with the maximum revenue-to-cost ratio, and embed G^v according to the embedding;

B. Embedding multiple virtual networks

In this subsection we deal with embedding a set $\mathcal{G}(i,t)$ of virtual networks to G^s by adopting a greedy strategy. To be specific, we first choose a virtual network $G^{v_1} \in \mathcal{G}(i,t)$ that leads to the maximum revenue-to-cost ratio and embed G^{v_1} to G^s. Let $\mathcal{G}_2(i,t) = \mathcal{G}(i,t) - \{G^{v_1}\}$. We then choose the next virtual network with the maximum revenue-to-cost ratio from $\mathcal{G}_2(i,t)$. This procedure continues until $\mathcal{G}_k(i,t)$ becomes empty or none of the virtual networks can be embedded. The algorithm is described in **Algorithm 3**.

Algorithm 3: *Embedding a set of virtual networks with given periodic resource demands*

Input: A set of virtual networks $\mathcal{G}(i,t)$ at tth time slot of interval i, the substrate network G^s, and the periodic resource demands of each virtual network $G^v \in \mathcal{G}(i,t)$
Output: Identify the set of virtual network requests that are implemented in $\mathcal{G}(i,t)$

while $\mathcal{G}(i,t) \neq \emptyset$ *or not all virtual networks are marked as unable to be embedded* **do**
  Let G^v_{max} be the virtual network that achieves the maximum revenue-to-cost ratio;
  Let η_{max} be the maximum revenue-to-cost ratio;
  for *each virtual network* $G^v \in \mathcal{G}(i,t)$ **do**
    Embedding G^v by **Algorithm 2**;
    if G^v *is rejected by* **Algorithm 2 then**
      Mark G^v as unable to be embedded;
    end
    else
      Let $\eta(G^v,i)$ be the achieved revenue-to-cost ratio through embedding of G^v;
      if $\eta_{max} < \eta(G^v,i)$ **then**
        $\eta_{max} \leftarrow \eta(G^v,i)$;
        $G^v_{max} \leftarrow G^v$;
      end
    end
  end
  Embed G^v_{max} to G^s by **Algorithm 2**;
  $\mathcal{G}(i,t) = \mathcal{G}(i,t) - \{G^v_{max}\}$;
end

V. Algorithm without periodic resource demands

The proposed algorithm so far assumed that the periodic resource demands of each virtual network are given in advance. In reality, very few users of virtual networks have such knowledge, this implies that the cloud service provider should be able to predict the periodic resource demands of virtual networks. In the following, we propose an algorithm to predict such the periodic resource demands by analyzing the historic resource demands of each embedded virtual network.

The basic idea behind the proposed algorithm is to embed a newly-arrived virtual network request according to its maximum resource demands initially, using **Algorithm 2**. It then adjusts the resources allocated to it after a number of intervals. Let $K(G^v)$ be the number of time intervals that the resource demands of G^v are adjusted. The amount of resource allocated to G^v will be adjusted every $K(G^v)$ intervals until the expiration of its duration $\tau(G^v)$, which means that there are $\lfloor \frac{\tau(G^v)}{K(G^v)} \rfloor$ adjustments to its demanded resources of G^v.

The key is how to adjust the amount of resource allocated to G^v. To this end, we keep a record of the actual resource demands of G^v for its past $K(G^v)$ time intervals, and predict its resource demands at the current time interval i. We then allocate the amount of resource to G^v that equals the predicted value. Recall that $\hat{C}(n^v,i,t)$ is the predicted computing resource demand of virtual node n^v. It has been shown that such resource demands follow a linear regression process [17]. We thus derive $\hat{C}(n^v,i,t)$ by an autoregressive moving average prediction method as follows,

$$\hat{C}(n^v,i,t) = \sum_{k=1}^{K(G^v)} \beta_{i-k} C(n^v, i-k, t), \tag{12}$$

where β_{i-k} is a given constant related to the resource demands at interval $i-k$, and $\sum_{k=1}^{K(G^v)} \beta_{i-k} = 1$ with $\beta_{i-k} \geq \beta_{i-k-1}$, $0 < \beta_{i-k} < 1$. This prediction gives an insight that the resource demands of a virtual network at the current time interval are related to its demands in its previous $K(G^v)$ intervals. The embedding algorithm is described in **Algorithm 4**.

Algorithm 4: *Embedding a set of virtual networks without given periodic resource demands*

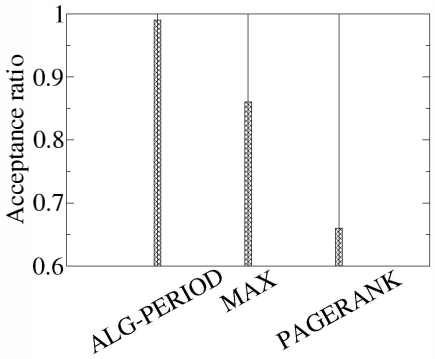
Input: Given a set $\mathcal{G}(i,t)$ of virtual networks, the substrate network G^s and the number of time intervals that the resource demand of each virtual network is adjusted, i.e., $K(G^v)$ for each virtual network $G^v \in \mathcal{G}(i,t)$
Output: Virtual network embeddings of requests in $\mathcal{G}(i,t)$

Embed each virtual network $G^v \in \mathcal{G}(i,t)$ according to its maximum resource demands by **Algorithm 1**;
for *each virtual network* G^v_{emd} *that has been embedded into* G^s **do**
  Let i' be the time interval that G^v was embedded in G^s;
  if G^v_{emd} *has stayed in* G^s *for* $\lceil \frac{i-i'}{K(G^v)} \rceil \cdot K(G^v)$ *intervals* **then**
    By Eq. (12), predict the computing resource demand of each virtual node n^v, $\hat{C}(n^v,i,t')$, and the bandwidth resource demand of each virtual link e^v, $\hat{B}(e^v,i,t')$, for current time interval i, where $1 \leq t' \leq T$;
    Reserve $\hat{C}(n^v,i,t)$ amount of computing resource for n^v, and $\hat{B}(e^v,i,t)$ amount of bandwidth resource for e^v in next $K(G^v)$ time intervals;
  end
end
/* at the end of interval i */
If $t = T$, log the actual resource demand of each embedded virtual network, and update the statistic information of the actions selected at the beginning of interval i;

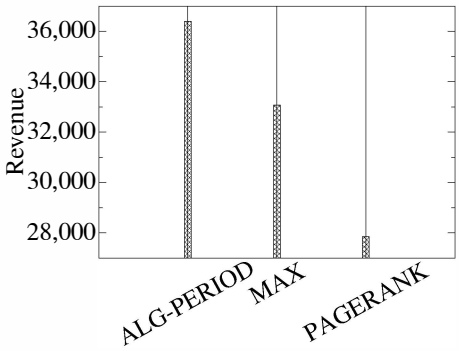
VI. Experimental Study

A. Simulation settings

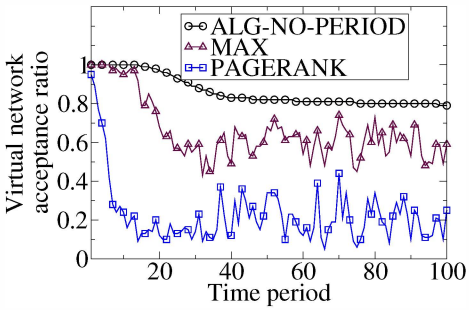
The substrate network G^s is generated by applying GT-ITM tool [12], which consists of 50 substrate nodes and there is an edge between each pair of nodes with a probability of 0.1, following similar settings in [8], [22], [27]. The capacity of each substrate node is randomly assigned a value in the range of $[2,000,\ 5,000]$ units (GHz) of computing resources, and the bandwidth capacity of each substrate link is in the range of $[10,\ 1,000]$ units (Mbps) [1], [13]. The number of virtual nodes of each virtual network varies from 2 to 10, and there is a virtual link between every two virtual nodes with a probability of 0.5. The amounts of peak computing and bandwidth demands of each virtual network are randomly generated from 2 to 10. The off-peak resource demands are no more than 80% of the peak resource demands. Parameters a and b in the embedding metric, Eq. (4), are set to 5, which will be explained later. The number of time slots of each interval is $T = 7$, e.g., 7 days a week. The arrival of virtual network requests follow the Poisson process with an average rate of 5 virtual networks per time slot, and the duration



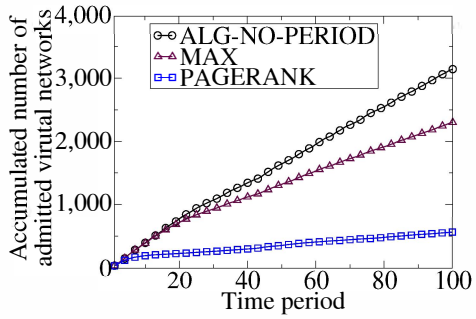
(a) The average acceptance ratio over 100 time slots of algorithm ALG-PERIOD



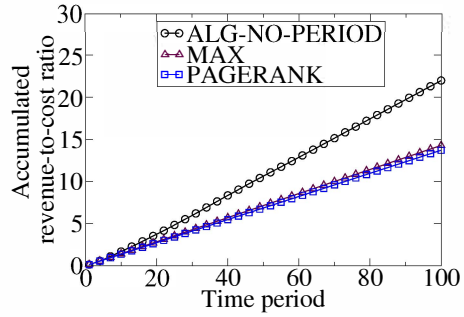
(b) The average revenue of embedding the virtual networks over 100 time slots of algorithm ALG-PERIOD



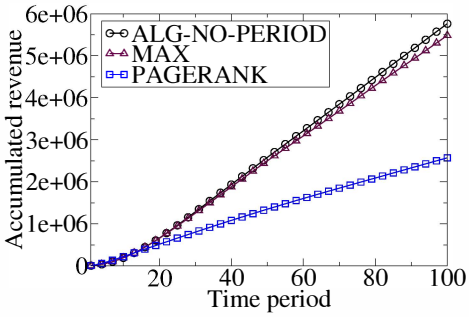
(c) The acceptance ratio of algorithm ALG-NO-PERIOD



(d) The accumulated number of admitted virtual networks of algorithm ALG-NO-PERIOD



(e) The accumulated revenue-to-cost ratio of algorithm ALG-NO-PERIOD



(f) The accumulated revenue of algorithm ALG-NO-PERIOD

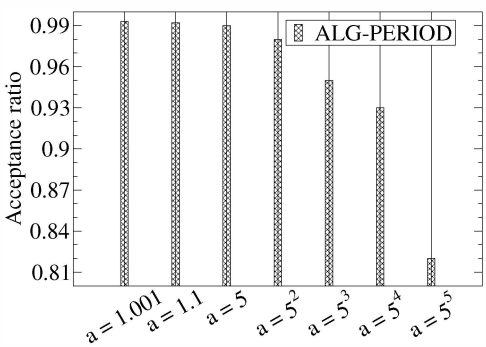
Fig. 3. The performance of algorithms ALG-PERIOD, ALG-NO-PERIOD, MAX and PAGERANK

of each virtual network varies with no more than 50 time intervals. The length of $K(G^v)$ in **Algorithm 4** is set to 5. As the actual resource demands by different virtual networks are different, virtual networks are classified into three categories: (1) growing networks whose resource demands increase by 10% with a probability of 0.3 per interval; (2) shrinking networks whose resource demands decrease by 5% with probability 0.3; and (3) stable networks with stable resource demands that never change. Unless otherwise specified, the default parameter settings will be adopted in our simulation.

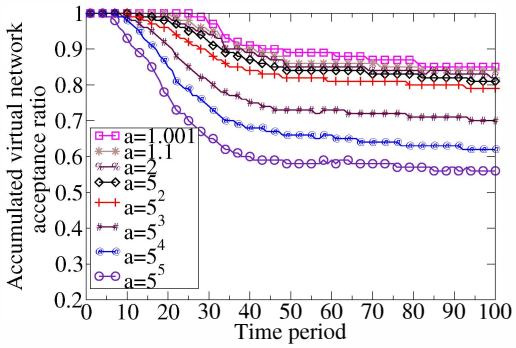
We evaluated the proposed algorithms for the VNE problem against two state-of-the-art existing algorithms. The first one is the algorithm in [27] in which the embedding ability of each substrate node n^s is the amount of available computing resource in n^s, i.e., $\Phi(n^s) = P(n^s, i, t)$, it embeds virtual networks to G^s according to their maximum resource demands, and we refer to it as algorithm MAX. The second one is the PageRank-based embedding algorithm [6], which first ranks virtual nodes and substrate nodes by adopting the PageRank algorithm, and then embeds each virtual network by mapping its virtual nodes, followed by embedding virtual links to the substrate network, which is referred to as PAGERANK. For simplicity, we use algorithms ALG-PERIOD and ALG-NO-PERIOD to denote **Algorithm 3** and **Algorithm 4** with and without the knowledge of periodic resource demands, respectively.

B. Performance evaluation

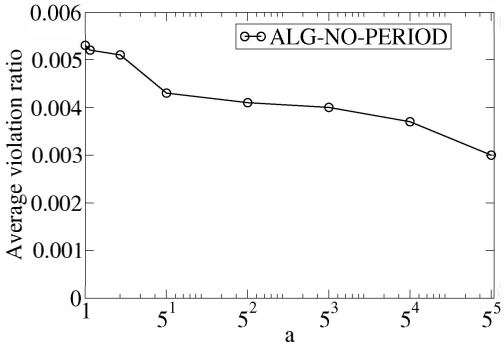
We first evaluate algorithms ALG-PERIOD and ALG-NO-PERIOD against algorithms MAX and PAGERANK in the time of 100 time slots. For the sake of diversity, we evaluate both average and accumulated performance over the 100 time slots. Fig. 3(a) shows that on average algorithm



(a) The impact of a on the acceptance ratio of algorithm ALG-PERIOD



(b) The impact of a on the acceptance ratio of algorithm ALG-NO-PERIOD



(c) The impact of a on the average violation ratio of algorithm ALG-NO-PERIOD

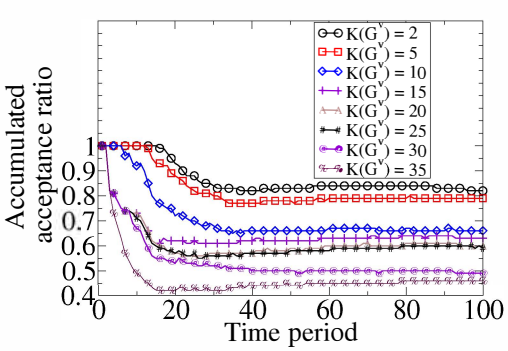
Fig. 4. The impact of a on the performance of algorithms ALG-PERIOD and ALG-NO-PERIOD

ALG-PERIOD admits around 15% and 30% more requests than algorithms MAX and PAGERANK do, respectively. Also, it can be seen from Fig. 3(b) that algorithm ALG-PERIOD brings 10% and 31% more revenues than algorithms MAX and PAGERANK do. The reason behind is that algorithm ALG-PERIOD adopts a fine-grained resource allocation by exploring the periodic resource demands, and a novel embedding metric by considering the dynamic workloads of the substrate network, whereas both MAX and PAGERANK do not. Furthermore, algorithm PAGERANK is the worst one among the three of them. Figures 3(c) and 3(d) clearly depict that algorithm ALG-NO-PERIOD consistently delivers the highest acceptance ratio and largest admitted number of virtual network requests among the three algorithms. From Figure 3(c), we can see that the virtual network acceptance ratios of algorithms MAX and PAGERANK oscillate a lot. The reason is that the number of admitted virtual networks are only influenced by the amount of available resources in the substrate network. The ratios fluctuate because of the oscillating amount of available resources caused by virtual network arrivals and departures. In addition, it can be seen from Fig. 3(e) that algorithm ALG-NO-PERIOD has a higher revenue-to-cost ratio than these of algorithms MAX and PAGERANK. Also, the revenue delivered by it is much higher than that by algorithms MAX and PAGERANK, as shown in Fig. 3(f). Notice that PAGERANK and MAX deliver similar revenue-to-cost curves in Figure 3(e) as they both are topology-aware embedding algorithms that avoid mapping virtual links onto long substrate paths that may cause more resource consumption.

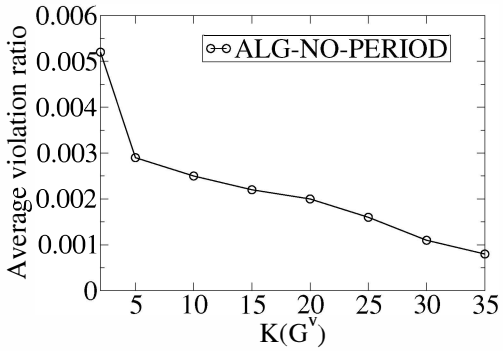
C. Impact of parameters

We then study the impact of parameters a and b in the embedding metric (Eqs. (4) and (5)) on the performance of the proposed algorithms ALG-NO-PERIOD and ALG-PERIOD, by varying a (=b) from 1.001 to 5^5. Figures 4(a) and 4(b) imply that the larger the value of a, the lower the acceptance ratio will be. The reason of this is that a larger a means that each substrate node is very likely reluctant to accept a virtual node when its utilization rate reaches one as this probably leads to SLA violations. Specifically, by Eq. (4), a larger a means that the embedding ability of a substrate node is more inclined to be determined by the exponential part, $a^{\frac{P(n^s,i,t)}{C(n^s)}}$, thereby underestimating some substrate nodes with high utilization ratio but large amounts of available resources. Fig. 4(c) plots the resource violations by algorithm ALG-NO-PERIOD with different values of a. It can be seen that algorithm ALG-NO-PERIOD has the highest violation ratio when $a = 1.001$, and the lowest violated ratio when $a = 5^5$. The algorithm delivers a violation ratio that is lower than 0.5% when $a = 5$, but gets a relative high acceptance ratio. This is the reason that we set $a = 5$ in our default parameter setting.

We finally investigate the impact of resource adjustment interval $K(G^v)$ on the algorithm performance by varying $K(G^v)$ from 2 to 35. Fig. 5 illustrates that a smaller value of



(a) The accumulated acceptance ratio



(b) The accumulated number of admitted virtual networks

Fig. 5. The impact of $K(G^v)$ on the performance of algorithm ALG-NO-PERIOD

$K(G^v)$ leads to a larger acceptance ratio but a higher violation ratio. For example, the average violation ratio of all admitted virtual network requests is 0.52% when $K(G^v) = 2$, and drops under 0.5% when $K(G^v)$ approaches to 5. The reason behind is that a smaller $K(G^v)$ means that more virtual networks will experience resource adjustments, which probably will lead to larger violation ratios as an inaccurate adjustment may violate the resource demand. Notice that the average violation ratio is below 0.5% when $K(G^v) = 5$ while the accumulated request acceptance ratio is still relatively high, which is also the reason that we set $K(G^v) = 5$ in our default parameter setting.

VII. Conclusion

In this paper we considered the virtual network embedding problem in a substrate network by developing novel embedding algorithms through incorporating a novel embedding metric, assuming that the periodic resource demands of each virtual network are given; otherwise, we devised an algorithm to predict the resource demands of virtual networks. We also evaluated the performance of the proposed algorithms through experimental simulations, and experimental results demonstrate that the proposed algorithms are promising and outperform existing heuristics.

Acknowledgement

The researches of Zichuan Xu and Qiufen Xia are partially supported by China Scholarship Council (CSC).

References

[1] H. Ballani et al. Towards predictable datacenter networks. *Proc. ACM SIGCOMM*, 2011.
[2] M. Bari et al. Data center network virtualization : a survey. *Communications Surveys &Tutorials*, Vol. 15, No. 2, pp. 909–928, IEEE, 2013.
[3] R. Bradford et al. Live wide-area migration of virtual machines including local persistent state. *Proc. ACM VEE*, 2007.
[4] N. F. Butt, M. Chowdhury, and R. Boutaba. Topology-awareness and reoptimization mechanism for virtual network embedding. *Proc. IFIP NETWORKING*, Springer, 2010.
[5] Z. Cai et al. Virtual network embedding for evolving networks. *Proc. IEEE GLOBECOM*, 2010.
[6] X. Cheng et al. Virtual network embedding through topology-aware node ranking. *ACM SIGCOMM Computer Communication Review*, Vol. 41, No. 2, pp. 38–47, 2011.
[7] N. M. M. K. Chowdhury and R. Boutaba. A survey of network virtualization. *Computer Networks*, Vol. 54, pp. 862–876, Elsevier, 2010.
[8] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba. Virtual network embedding with coordinated node and link mapping. *Proc. IEEE INFOCOM*, 2009.
[9] I. Fajjari et al. Vnr algorithm: a greedy approach for virtual networks reconfiguration. *Proc. IEEE GLOBECOM*, 2011.
[10] A. Fischer et al. Virtual network embedding: a survey. *Communications Surveys & Tutorials*, Preprint, IEEE, 2013.
[11] D. Gmach et al. Workload analysis and demand prediction of enterprise data center applications. *Proc. IEEE IISWC*, 2007.
[12] GT-ITM. http://www.cc.gatech.edu/projects/gtitm/.
[13] C. Guo et al. SecondNet: a data center network virtualization architecture with bandwidth guarantees. *Proc. ACM CONEXT*, 2010.
[14] I. Houidi et al. Virtual network provisioning across multiple substrate networks. *Computer Networks*, Vol. 55, pp.1011–1023, Elsevier, 2011.
[15] I. Houidi, W. Louati, and D. Zeghlache. A distributed and autonomic virtual network mapping framework. *Proc. IEEE ICAS*, 2008.
[16] I. Houidi et al. Adaptive virtual network provisioning. *Proc. ACM VISA*, 2010.
[17] A. Kansal et al. Virtual machine power metering and provisioning. *Proc. ACM SoCC*, 2010.
[18] J. Lischka and H. Karl. A virtual network mapping algorithm based on subgraph isomorphism detection. *Proc. ACM VISA*, 2009.
[19] P. Padala et al. Adaptive control of virtualized resources in utility computing environments. *Proc. ACM EUROSYS*, 2007.
[20] G. Sun et al. Adaptive provisioning for evolving virtual network request in cloud-based data centers. *Proc. IEEE GLOBECOM*, 2012.
[21] U.S. Department of Energy. Creating energy efficient data centers. 2007.
[22] M. Yu et al. Rethinking virtual network embedding: substrate support for path splitting and migration. *ACM SIGCOMM Computer Communication Review*, Vol. 38, No. 2, pp. 17–29, 2008.
[23] S. Zhang et al. An opportunistic resource sharing and topology-aware mapping framework for virtual networks. *Proc. IEEE INFOCOM*, 2012.
[24] S. Zhang and X. Qiu. A novel virtual network mapping algorithm for cost minimizing. *Journal of Selected Areas in Telecommunications*, Vol. 02, No. 01, pp. 1–9, 2011.
[25] S. Zhang, J. Wu, and S. Lu. Virtual network embedding with substrate support for parallelization. *Proc. IEEE GLOBECOM*, 2012.
[26] Y. Zhou et al. Incremental re-embedding scheme for evolving virtual network requests. *IEEE Communication Letters*, Vol. 17, No. 5, pp. 1016–1019, 2013.
[27] Y. Zhu and M. Ammar. Algorithms for assigning substrate network resources to virtual network components. *Proc. IEEE INFOCOM*, 2006.