Toward efficient dynamic virtual network embedding strategy for cloud networks

Duc-Lam Nguyen, Hyungho Byun, Naeon Kim and Chong-Kwon Kim

Abstract
Network virtualization is one of the most promising technologies for future networking and considered as a critical information technology resource that connects distributed, virtualized cloud computing services and different components such as storage, servers, and application. Network virtualization allows multiple virtual networks to coexist on the same shared physical infrastructure simultaneously. One of crucial factors in network virtualization is virtual network embedding which provisions a method to allocate physical substrate resources to virtual network requests. In this article, we investigate virtual network embedding strategies and related issues for resource allocation of an Internet provider to efficiently embed virtual networks that are requested by virtual network operators who share the same infrastructure provided by the Internet provider. In order to achieve that goal, we design a heuristic virtual network embedding algorithm that simultaneously embeds virtual nodes and virtual links of each virtual network request onto the physic infrastructure. Via extensive simulations, we demonstrate that our proposed scheme significantly improves the performance of virtual network embedding by enhancing the long-term average revenue as well as acceptance ratio and resource utilization of virtual network requests compared to prior algorithms.

Keywords
Cloud computing, virtual network embedding, network virtualization, resource allocation

Introduction
Recently, cloud computing has been positioned as a large-scale distributed model for providing on-demand information technology (IT) resources such as storage, servers, networks, and application as a service through data centers or high-speed networks. Cloud computing centers are increasingly facilitated with virtualization technologies to make better use of provider’s resources and to guarantee higher flexibility and reliability. The concept of network virtualization (NV) has provided a promise for the current and future infrastructure as a service (IaaS) provision in cloud networks. NV has become a popular concept in a wide range of technologies. NV enables network operators to share their physical infrastructure with different tenants by forming multiple virtual networks (VNs).

In the cloud environment of NV, the Internet service provider (ISP) is classified into the infrastructure providers (InPs) and service providers (SPs). InPs build, maintain, and manage the physical infrastructure, while service providers are owners of VNs. SPs deploy different protocols as well as offer multiple end-to-end heterogeneous services to the end user.
NV brings many potential benefits. First, it offers a virtual environment to design and evaluate new network architectures or protocols. Second, it enables resource sharing among VNs and optimizes resource allocation of physical infrastructure. In NV, a VN is a combination of virtual nodes and virtual links on top of a substrate network (SN). A virtual node can be hosted on any available substrate node and interconnected by a set of virtual links. Virtual links can be spanned over several substrate links, forming a virtual topology. SPs send virtual network requests (VNRs) with the information of virtual nodes and virtual links capacity requirements to InPs which map the VNRs onto SN in turn, as shown in Figure 1. In cloud infrastructures, VNs are offered as a service to interconnect distributed cloud components and resources to support a plethora of cloud-based services for customers. As a result, the cloud service request can be abstracted as a VNR where virtual nodes stand for virtual machines and virtual links stand for cloud infrastructure links. VN provision can be realized via cloud systems such as cloud stack or open stack using software-defined network controllers and underlying data plan system OpenvSwitch.

A major challenge of the NV technology is known as a virtual network embedding (VNE) problem that deals with the efficient allocation of substrate resources to virtual requests. An efficient utilization of SN resources depends on the VNE algorithms that coordinates node and link resource constraints. The node constraints consist of several factors such as CPU performance, memory, and storage capacity or even the location of nodes in the substrate topology. The link constraints can be bandwidth, bit error rate, delay, throughput, jitter, and so on. Consequently, the VNE can be divided into two stages: virtual node mapping which assigns the virtual nodes to the substrate nodes and virtual link mapping where the virtual links are mapped onto physical links connecting these virtual nodes after the virtual node mapping. It is well-known that the VNE problem is non-deterministic polynomial-time (NP)-hard\textsuperscript{5–7} and the problem can be reduced to multi-way separator problem.\textsuperscript{8}

Recently, the VNE problem has received significant research attentions that aim to design diversified Internet architectures. The primary goal of VNE study is to maximize the economic benefit (called long-term average revenue) by accommodating as many VNRs as possible on a given physical network.\textsuperscript{9–12} However, the revenue of InPs depends on the VNRs, and in real-world situations, VNRs are dynamic and unpredictable. It is known as the problem with online VNRs. However, most prior works have just focused on static VNRs. Also, most of previous researches separated the VNE problem into two independent phases. That is, the virtual nodes are mapped onto substrate nodes at the first phase according to the local resource of substrate nodes without caring the other attributes such as location of substrate nodes or available bandwidth between nodes while these parameters have significant influence on the VNE algorithms’ performance. Then, the virtual links are mapped simply using the shortest path algorithm.\textsuperscript{9} However, the separation of virtual node mapping and virtual link mapping might lead neighboring virtual nodes to be separated widely in the

![Figure 1. Virtual network embedding problem.](image)
physical topology. Besides, this separation increases the cost of virtual link mapping phase. So, the acceptance ratio, as well as long-term revenue, might decreases.

With the aforementioned motivations, in this article, we investigate a new strategy to coordinate virtual link mapping and virtual node mapping in terms of handling online VNRs. The overview of our strategy is shown in Figure 2, and the arriving VNRs are handled within scan-time intervals; if there is any VNR which cannot be processed within a scan-time window, it is deferred to try in the next scan-time windows. In order to maximize the long-term revenue of InPs, we propose a new strategy to handle VNRs in each scan-time window. VNRs are grouped, analyzed, and ordered following CPU capacity and bandwidth capacity demand. We also consider the resources of the SN simultaneously to optimize resource allocation. The benefit of proposed algorithm lies in the fact that it can further leverage the dynamic VNRs and thus increase the revenue of InPs. The key idea of our algorithm is that it first chooses X embedding candidates where X is the search number, and in each candidate embedding, we choose a substrate node as the parent node (root node) which is mapped to the first VN node which is chosen by a selection rule. Then, VN nodes and links are mapped simultaneously based on joint node and link constraint. Finally, we choose the one with the best quality and which is feasible for embedding. The details of proposed algorithm are described in the later section. Through extensive simulations, we demonstrate that our proposed algorithm outperforms the prior algorithms.

The main contributions of this article can be summarized as follows:

1. We find out the limitation of existing schemes. Without the ability to adapt dynamic VNRs and the separation in virtual node and link mapping phases, the existing schemes may perform inefficiently in case of dynamic and large-scale networks.
2. A new VNE algorithm is proposed. We introduce a new real-time VNE strategy which coordinates virtual node and virtual link mapping while handling dynamic VNRs as in the real-world cloud environment. The key point is that the algorithm handles a group of VNRs in each scan-time window.
3. Through extensive experiments, we show that the proposed scheme achieves a significant improvement compared to prior algorithms in terms of acceptance rate, resource utilization, and long-term average revenue.

The rest of this article is organized as follows: related works and issues are discussed in section “Related works.” In section “System modeling,” we formulate the VNE problem. In section “VNE algorithm,” we present our real-time VNE algorithm. Then, we evaluate our proposed VNE algorithm in section “Performance evaluation.” Finally, section “Conclusion” concludes of this article.

Related works
The VNE problem has received significant research attentions in both academic and industrial areas and presented the main resource allocation challenge in virtual cloud infrastructure. There are many studies to find out the efficient strategy to maximize the benefit.
for InPs including subgraph isomorphism,\(^\text{12}\) path splitting and migration,\(^\text{9}\) topology-aware node ranking\(^\text{13}\) or other heuristic algorithms, and meta-heuristic algorithms.\(^\text{14–16}\) Generally, the existing studies can be divided into two primary categories: online approaches and offline approaches.

The offline algorithm processes are based on the assumption that all VNRs are known and defined in advance. Beck et al.\(^\text{17,18}\) proposed a distributed and parallel VNE (DPVNE) framework, which can be used in combination with various cost-reducing embedding algorithms. Through evaluation results, they proved that DPVNE achieves smaller message overhead and suggests a trade-off between message overhead and parallelism level. Fajjari et al.\(^\text{19}\) introduced a Max-Min Ant Colony Meta-heuristic algorithm which uses parallel artificial ants to explore the search space iteratively for feasible VNR candidates and select the candidates based on several specific resource constraints. The work of J Lu and Turner\(^\text{20}\) considered the single VNE in specific backbone-star topologies and assumed that the substrate resources are unlimited with only bandwidth constraints. In these offline VNE approaches, the VNR can be considered to gather for embedding, which may achieve a better performance compared to online approaches. However, offline VNE approaches do not contemplate the possibility of re-mapping existing VNRs to enhance the performance of the embedding SN. The arrival of VNR is unpredictable in a real environment, where dynamic algorithms are more suitable and preferred.

The dynamic VNE approaches are about the on-demand VNR assignment problem where algorithms dynamically compute a feasible set of substrate nodes and links to embed the virtual nodes and links upon the arrival of VNRs. Chowdhury et al.\(^\text{11}\) proposed a coordinate node and link mapping strategy by formulating the VNE as a mixed integer programming (MIP), in which the node mapping is achieved by facilitating the link mapping in the subsequent phase. Randomized and deterministic rounding techniques are used to guarantee better correlation between node and link mapping phases. Yu M et al.\(^\text{9}\) proposed the rethinking design of the SN to enable simpler mapping algorithms and more efficient use of physical resources, without restricting the problem space. They first designed the SN to split a virtual link to span over multiple substrate paths. They employed the path migration to periodically re-optimize the utilization of the SN to support the accommodating of new requests. In this research, a virtual link can be mapped to multiple substrate paths for splittable flows, and path migration is used to optimize the utilization of physical resources periodically. X Cheng et al.\(^\text{21}\) handled VNE problem using the PageRank algorithm, where both substrate nodes and virtual nodes are ranked based on metrics, and then, the VN is embedded based on these ranks. The goal is to increase the acceptance ratio of online VNRs as well as overall revenue.

However, Ricci et al.\(^\text{5}\) provided a heuristic algorithm based on a centralized algorithm which tends to maintain low and balanced substrate nodes and substrate links. A distributed VN mapping algorithm was proposed by Houdi et al.\(^\text{22}\). In this research, the VNRs can be decomposed into hub-and-spoke clusters, and these clusters can be embedded independently, so it achieves to reduce the complexity of VNE. However, this approach has lower performance and it was efficiently compared to centralized approaches.

Besides, in cloud environments, the VNE problems also subject to dynamic changes and have to deal with node and link failures.\(^\text{23–26}\) Houidi et al.\(^\text{23}\) and Gou et al.\(^\text{24}\) proposed adaptive and survivable embedding algorithms to deal with link failures where the objective is to minimize the embedding cost. There are several types of failures such as physical node failure, physical link failure, and virtual link failure.

### System modeling

In this section, we present the general embedding problem. Notations used in this context are summarized in Table 1.

#### Network models

**SN.** A physical can be described as a weighted undirected graph \(G^S = (N^S, L^S)\), where \(N^S\) is the set of substrate nodes and \(L^S\) is the set of substrate links. Each substrate link \(P(n^S, n^S) \subset L^S\) interconnects two substrate nodes \(n^S\) and \(n^S\). We denote the set of loop-free substrate paths by \(P^S\). The \(C^S(n^S)\) and \(BW^S(P^S)\) refer to

| Notation | Description |
|----------|-------------|
| \(C^S\)  | Substrate network |
| \(N^S\)  | Set of substrate nodes |
| \(L^S\)  | Set of substrate links |
| \(G^V\)  | Virtual network requests |
| \(N^V\)  | Set of virtual nodes |
| \(L^V\)  | Set of virtual links |
| \(R^S\)  | Physical node resources |
| \(R^L\)  | Physical link resources |
| \(C_{\text{Req}}(n^V)\) | CPU request of VN node \(n \in N^V\) |
| \(C_{\text{Rem}}(n^V)\) | CPU available remaining of SN node \(n^S \in N^S\) |
| \(C_{\text{Rem}}^S(n^S)\) | VN-node-capacity |
| \(C_{\text{Rem}}^L(n^V)\) | SN-node-capacity |
| \(Ltr^V(\vec{P}^V)\) | Link-traffic-ratio |
| \(BW_{\text{Req}}^L(\vec{P}^L)\) | BW requested for VN link \(\vec{P}^L \in L^V\) |
| \(BW_{\text{Rem}}^L(\vec{P}^L)\) | Available BW of SN link \(\vec{P}^L \in L^S\) |

VNE: virtual network; SN: substrate network; BW: bandwidth.
total available CPU capacity of substrate node $n_s$ and total available bandwidth of substrate link $l^s$, respectively. An example of SN is shown in lower part of Figure 1, where the numbers in a square are the attribute of nodes indicated by the available CPU capacity. The numbers beside the edges are the attribute of links indicated by the available bandwidth capacity.

VNR. Similar to the SN, we denote a VNR by $G^V = (N^V, L^V)$, in which $N^V$ is a set of virtual nodes and $L^V$ is a set of virtual links. $C_{\text{req}}^{V}(n^V)$ and $BW_{\text{req}}^{V}(l^V)$ are CPU capacity and bandwidth capacity requirement of virtual nodes and virtual links, respectively. For example, as VNRs are shown in upper part of Figure 1, the numbers in a square are the attribute of virtual nodes indicating the CPU capacity demands and the numbers beside the edges are the attribute of virtual links indicating the bandwidth demands.

VNE problems

The VNE problem is defined by an embedding $\Gamma$ from a VN $G^V$ to a subset of the SN $G^S$. The embedding can be expressed as follows

$$\Gamma : G^V \rightarrow (N^S, P^S, R^N, R^L)$$

where $N^S \subset N^S$ and $P^S \subset P^S$ and $R^N$ and $R^L$ are physical node and link resources allocated for VNRs. The VNE problem can be divided into two categories: (1) virtual node mapping and (2) virtual link mapping. First, each virtual node of a VNR is mapped to SN as follows

$$\Gamma^N : N^V \rightarrow N^S$$

where $\Gamma^N(n^V) \in N^S$ with $n^V \subset N^V$ subject to the CPU capacity constraint as follows

$$C_{\text{req}}^{V}(n^V) \leq C^S(\Gamma^N(n^V))$$

$$C_{\text{rem}}^{S}(n^S) = C(n^S) - \sum_{n^V \subset N^V} C_{\text{req}}^{V}(n^V)$$

where $C_{\text{req}}^{V}(n^V)$ is the amount of CPU capacity requested by the virtual node $n^V$ and $C_{\text{rem}}^{S}(n^S)$ is the remaining CPU capacity available of substrate node $n^S$ after embedding virtual node $n^V$. Second, each virtual link can be mapped to paths connecting the corresponding nodes of SN. We denote virtual link mapping as follows

$$\Gamma^L : L^V \rightarrow P^S$$

where $\forall l^V(n^V_i, n^V_j) \subset L^V$, and $P^S(\Gamma^N(n^V))$ is the set of substrate path connecting virtual node $n^V_i$ and $n^V_j$ ($\Gamma^N(n^V_i) \subset \Gamma^S(l^V_i)$) subject to bandwidth capacity constraints

$$BW_{\text{req}}^{V}(l^V) \leq BW^S(p), \forall p \in \Gamma^L(l^V)$$

$$BW^S(p) = \min_{l^V \subset L^V} BW_{\text{req}}^{V}(l^V)$$

$$BW_{\text{rem}}^{S}(l^V) = BW^S(l^V) - \sum_{l^V \subset L^V} BW_{\text{req}}^{V}(l^V)$$

where $BW_{\text{rem}}^{S}(l^V)$ is the requested bandwidth capacity for virtual link $l^V \subset L^V$. $BW_{\text{rem}}^{S}(l^V)$ denotes the amount of available bandwidth capacity remaining of substrate link $l^V$.

Performance metrics

In this article, the following metrics are used to evaluate the performance of our proposed algorithm. First, we define the long-term average revenue $R(G^V)$ where the Internet provider obtains from the SP for serving a VNR $g^V \subset G^V$ as follows

$$\text{Maximize} \lim_{t \rightarrow \infty} \frac{\int_{t_{i-1}}^{t_i} R_g(t)dt}{t_i - t_{i-1}}$$

where $k$ is the number of VNRs which are accepted successfully by SN in duration of the VNR being served in the SN. And $t_d = t_e - t_a$ with $t_a$ denoting the arriving time of a VNR and $t_e$ is an expiration time of the VNR.

In this article, we aim to maximize the long-term average revenue

$$R_g = \left[ \sum_{n^V \in N^V} C^{V}(n^V) + \sum_{l^V \in L^V} BW(l^V) \right] \times t_d$$

Second, in each time window, the ratio of the number of successful embedded virtual network to all VNRs is defined as Acceptance Ratio. The variable $x$ is in $\{0, 1\}$, if the $i$th VNR is accepted successfully, $x_i = 1$, otherwise $x_i = 0$

\[
Acceptance \ ratio = \frac{\sum_{i=1}^{k} x_i}{\sum_{i=1}^{k} |VNRs|} \times 100\% 
\]

Third, higher substrate resource utilization has always been an important objective. Resource utilization includes three different types: node utilization, link utilization, and network resource utilization

$$\text{NodeU} = \frac{\sum_{n^V \in N^V} C(n^V)}{\sum_{n^S \in N^S} C(n^S)}$$
which has highest required VN-Node-Capacity ing order of hop distance from the parent virtual node algorithm, and then, virtual nodes are sorted in ascend-
each VNR is detected using Breath-First Search (BFS)
total allocated bandwidth, delay and throughput, and
link mapping will be successful and be optimized.

VNE algorithm
In this section, we introduce our real-time VNE strategy. The overview of our proposed algorithm is shown in Figure 2, and the conjunction of virtual node-link mapping is depicted in detail.

Online VNR handling
In reality, the VNRs dynamically arrive and stay in SN for an interval time before leaving. In order to handle online VNRs, we divide time into scan-time windows (intervals) which consider how long the VNRs are executed frequently (i.e. an hour or several days). In detail, VNRs in each scan-time window are processed sequentially in each scan-time window.

Virtual node mapping
In this step, we perform the node mapping process which considers how to map virtual nodes on substrate nodes. Because node mapping affects link mapping directly, we want to perform node mapping first. Then, link mapping will be successful and be optimized.

Followings are the key idea of our node mapping algorithm to achieve this goal. First, we measure the CPU capacity and bandwidth resource of virtual nodes and substrate nodes. Then, we collect a group of X VNRs and process X VNRs sequentially in each scan-time window.

In terms of virtual nodes, when a VNR arrives, we need to determine the mapping sequence of virtual nodes to the available substrate nodes. We define the VN-Node-Capacity for virtual nodes $n^V$ as follows

\[
\text{Cap}^{VN}(n^V) = C_{\text{req}}^{V}(n^V) + \sum_{P' \in L(n)} BW_{\text{rem}}^{V}(P') \tag{10}
\]

First of all, over each time interval, the topology of each VNR is detected using Breath-First Search (BFS) algorithm, and then, virtual nodes are sorted in ascending order of hop distance from the parent virtual node which has highest required VN-Node-Capacity value (equation (10)). The parent virtual node is first embedded into SN due to difficulty in mapping the virtual nodes, which needs high-requirement CPU capacity, and these virtual nodes may have many corresponding virtual links. For example, we use Figure 3(a) to illustrate the process. That is, VN-Node-Capacity value of virtual node a is 6 + 4 + 5 = 15 unit, and the values of virtual node b and virtual node c are 9 and 10, respectively. Virtual node a has the highest VN-Node-Capacity value and virtual node a is the parent virtual node and the embedding sequence is $a \rightarrow c \rightarrow b$. Second, after choosing parent virtual node for each VNR, we determine a parent substrate node for each parent virtual node. Because we selected $X$ parent virtual nodes of X VNRs, $X$ parent substrate nodes also should be selected. We use evaluation metric SN-Node-Capacity value of parent substrate nodes which is similar to parent virtual nodes. And the substrate nodes are sorted in descending order of SN-Node-Capacity values

\[
\text{Cap}^{SN}(n^S) = C_{\text{req}}^{S}(n^S) + \Psi \sum_{P' \in L^S(n^S)} BW_{\text{rem}}^{S}(P') \tag{11}
\]

where $L^S(n^S)$ is the set of adjacent substrate links of substrate node $n^S$. In fact, there are many substrate nodes which have resources redundantly to become parent node. However, we prefer to select the parent node which has enough resources. As a prior study, we also do not use $C(n^S)$ alone as an evaluation metric, because we consider making sure that substrate node $n^S$ has enough CPU capacity available. Similarly, we use Figure 3(b) to illustrate the process, and the SN-Node-Capacity value of substrate node 3 is highest $(10 + 5 + 7 + 9 + 6 = 37)$, so that node 3 is the parent substrate node for VNR which has the parent virtual node a.
Virtual node-link mapping conjunction

Next, we explain how to embed virtual nodes and links simultaneously onto SN. First of all, we define a metric called \( \text{Link-traffic-ratio} \) \( \text{Ltr}(l_S) \) which implies the ratio between a substrate link \( l_S \) and a total bandwidth of all substrate links. The \( \text{Ltr}(l_S) \) represents the intensity of overall interference that the SN link \( l_S \) affects its other neighboring links. The \( \text{Link-traffic-ratio} \) of a substrate path is the sum of the \( \text{Link-traffic-ratio} \) of all links in the path

\[
\text{Ltr}(l_S) = \frac{\text{BW}(l_S)}{\sum_{l^S \in L^S} \text{BW}(l^S)}
\] (12)

Figure 3 shows an example of \( \text{Link-traffic-ratio} \). First of all, we assume that parent substrate node \( n_S \) and parent virtual node \( n_V \) of VNR \( g_V \) are selected as mentioned above. The rest of virtual nodes \( n^c \) are embedded into corresponding substrate nodes, and virtual links between virtual nodes \( n^c \) are mapped following the shortest paths in terms of \( \text{Link-traffic-ratio} \), which according to each path has to satisfy bandwidth requirements of virtual links. The node mapping rule is designed to minimize the load stress added by mapping new virtual nodes to substrate nodes. In fact, we aim to choose the reasonable substrate nodes and shortest path offering the best bandwidths in order to reduce resource allocated to substrate links. Selecting substrate nodes \( n^S \) for corresponding \( n^V \) is as follows

\[
n^S \in \arg \min \sum_{n^V_1 \in N^V, n^S_1 \in N^S} \text{BW}(n^V_1, n^S_1) \text{Ltr}(n^V_1, n^S_1)
\] (13a)

s.t. Bandwidth demands by added VN links (13b)

Equation (12) also shows the bandwidth capacity requirement ratio to map virtual link between a pair of virtual node \( n^V_1 \) and \( n^V_2 \). For example, we use Figure 3 to illustrate the process. In this example, we consider one VNR (Figure 3(a)), where \( a \) is the parent virtual node and 3 the parent substrate node. As mentioned, the next virtual node is node \( c \), and we need to find the corresponding substrate node which has the smallest \( \text{link-traffic-ratio} \) value from parent substrate node \( C \) and satisfy bandwidth requirement of virtual link \( (a,c) \), that is substrate node 5, so virtual link \( (a,c) \) is embedded to substrate link \( (3,5) \). Finally, we find the available substrate node to map virtual node \( b \).

Following equation (12), we calculate the bandwidth capacity requirement ratio from substrate nodes 3 and 5 to other substrate nodes 1, 2, and 4. In terms of substrate node 1: \( 2 \times 0.084 + 4 \times 0.15 = 0.768 \), for substrate node 2: \( 2 \times (0.13 + 0.1) + 4 \times 0.13 = 0.98 \), and for SN node 4: \( 2 \times 0.1 + 4 \times 0.11 = 0.64 \). So, we select...
substrate node 4 to map virtual node b, and virtual links \((b, c)\) and \((b, a)\) are embedded to substrate links \((4, 5)\) and \((4, 3)\), respectively. The process of our proposed algorithm called RT-VNE Algorithm is described in Algorithm 1. As VNR is embedded unsuccessfully, they then returned to queue and wait for next time interval (Figure 2).

**Performance evaluation**

In this section, we evaluate the performance of our proposed VNE algorithm in detail and compare to the prior algorithms under various scenarios. Our evaluation concentrates on quantifying the advantage of coordinate node and link mapping phases as well as proposed dynamic VNRs handling in terms of long-term average revenue and acceptance ratio. All simulations are performed on the server Intel® Core™ 2 Quad C Q9550 at 2.83 GHz (4Cs), 8GB memory, 1 TB disk and OS Linux 2.6.

**Simulation environment**

**SN.** Similar to most of the prior studies, we used the Georgia Tech Internet Topology Model (GT-ITM) tool\(^2\) to generate SN and VN topology. The SN is composed of 100 nodes, close to mid-size of an ISP scale. These substrate nodes are distributed geographically in a \((10 \times 10)\) grid square. Each pair of substrate nodes is connected with a probability of 0.5. The CPU capacity of nodes and bandwidth capacity of links in SN follow a uniform distribution from 0 to 300 unit.

**VNR.** We evaluate the performance of VNE algorithms in various cases of VNRs. In each VN, the number of virtual nodes is distributed uniformly from 2 to 10 nodes for a small size, 10 to 50 nodes for mid-size, and 50 to 80 nodes for a large size. The average of VN connectivity is set to be a probability of 0.5. Similar to SN configuration, the CPU capacity and bandwidth capacity of each virtual node and link are uniformly distributed between 0 and 30 units. The arrival of VNRs is followed by a Poisson process with an average which ranges from 2 to 14 VNRs per scan-time window. The lifetime of each VNR at the SN follows an exponential distribution with an average of five scan-time windows. In each simulation, there are approximately 500 scan-time windows, which correspond to around 2500 VNRs on average. The value of candidate VNRs \(X\) is varied from 2 to 10 (the default value is 9). Figure 4 shows the long-term average revenue as \(X\) (the number of searching candidates) increases from 2 to 10. We vary the number of substrate nodes with 20 nodes, 40 nodes, and 60 nodes. Figure 4 shows that with the value of \(X\) increases from 6 to 10, the average revenue achieves good results. In Figure 5, the computation time also increases sharply with the increasing number of SN nodes, because the number of substrate links increases due to the fixed SN density. The value of searching candidate \(X\) might depend on the SN and VN sizes. The result implies that a small search number out of the large search space may be sufficient enough. Figures 4 and 5 also demonstrate that our proposed algorithm is computationally tractable because the proposed algorithm generates the mapping results in the order of seconds which is reasonable in real-world environment.

**Comparison setup.** Our proposed algorithm is implemented in C/C++ and we compared our algorithm to previous state-of-art algorithms. To evaluate the performance of our proposed algorithm RT-VNE, we compare our algorithm to existing algorithms (Table 2) including greedy mapping algorithm,\(^9\) a cluster-center VNE algorithm,\(^6\) and three ViNE algorithms.\(^11\) In greedy mapping algorithm (G-SP), virtual nodes with
higher priority are mapped to substrate nodes with higher resources, and virtual links are embedded using \textit{k-shortest-path} algorithm without reconfiguration. And with the VNE algorithm CC-VNE,\textsuperscript{6} the authors first determine the cluster of substrate nodes which will serve virtual nodes. Then, they find out which substrate node has the highest available resource and designate that node as the cluster center. After that, the other substrate nodes are sorted based on the distance from selected substrate nodes. Finally, the virtual nodes are mapped into selected substrate nodes such that higher degree virtual nodes are embedded to corresponding higher neighborhood resource availability substrate nodes.

\textbf{RT-VNE algorithm performance}

\textit{Long-term average revenue.} We compare the proposed VNE algorithm RT-VNE to five existing algorithms including G-SP, CC-VNE, R-ViNE, D-ViNE, and D-ViNE-SP through 500 scan-time windows. In our scenarios, VNRs are divided into three groups: small-sized [2,10] nodes, medium-sized [10,50] nodes, and large-sized [50,80] nodes. From Figures 6–8, we find out several interesting observations. First, from the result in Figure 7, RT-VNE leads to better long-term average revenue than the existing algorithms in three cases which are small, medium, and large in size in the range of 500 scan-time windows. Along with these results, it implies that our proposed algorithm embeds VNRs that make more benefit revenue without taking care the size of VNRs (small, medium, or large) and increases the acceptance ratio of embedding. We believe that the reason is the higher acceptance ratio of RT-VNE (Figure 6) compared to existing algorithms as well as mapping virtual nodes to the substrate nodes with adequate resources increasing the probability of satisfying the resource requirements of the VNRs and consequently helping to accelerate convergence. Moreover, in addition to node mapping constraints, RT-VNE also considers the link bandwidth constraint which obviously makes link mapping phase easier. Thus, RT-VNE algorithm achieves a higher probability of accepting VNRs.

| Notation | Description |
|----------|-------------|
| G-SP\textsuperscript{9} | Greedy node mapping with shortest path–based link mapping |
| CC-VNE\textsuperscript{6} | Cluster-center VNE approach |
| RT-VNE | Proposed scheme |
| R-ViNE\textsuperscript{11} | Randomized node mapping with splittable link mapping using MCF |
| D-ViNE\textsuperscript{11} | Deterministic node mapping with splittable link mapping using MSF |
| D-ViNE-SP\textsuperscript{11} | Deterministic node mapping with shortest path–based link mapping |

VNE: virtual network embedding; MCF: multicommodity flow; MSF: multicommodity k-splittable flow.

\textbf{Figure 6. Acceptance ratio of algorithms.}

\textbf{Figure 7. Average revenue of algorithms.}

\textbf{Figure 8. Average revenue between algorithms on different VN scales.}
Acceptance ratio. As shown in Figure 6, the proposed algorithm, RT-VNE, leads to higher acceptance ratio compared to existing algorithms through a joint node and link mapping. The acceptance ratio of RT-VNE (over 80%) is stable and higher than others (G-SP is around 52%, CC-VNE is about 72%, and R-ViNE, D-ViNE, and D-ViNE-SP are around 76%, 78%, and 63%, respectively). Since the linear programming (LP) relaxation and rounding techniques used by \{R|D\}-ViNE algorithms may incur infeasible VNE solution, they suffer low performance in terms of VNR acceptance ratio and consequently result in lower long-term average revenue (Figures 7 and 8). G-SP and CC-VNE always select maximum substrate objects for embedding. Thanks to Link-traffic-ratio, and the effect of combining node and link mapping is magnified when the available link resources are not enough. However, the influence of link interferences on embedding is not changed significantly since the relative among resources remains the same. Also, VN consolidation increases the efficiency of SN resource usage and saves more available resources for other future VNRs.

Resource utilization. The average resource utilization of substrate nodes and substrate links from different VNE algorithms are depicted in Figures 9 and 10. In terms of small, medium and large scales, the values of node utilization are around 20%, 38%, and 58%, respectively. The values of link utilization are approximately 20%, 40%, and 79%, respectively. Thanks to Link-traffic-ratio, even the acceptance ratio of RT-VNE is higher than others, and the average node and link utilization of RT-VNE are quite the same with the others. Consequently, network utilization is enhanced. As the VN scale increases, all compared algorithms have the higher node and link utilization. This is because there are more substrate nodes which have to be active to handle incoming VN nodes as well as the number of SN links increases when VN scale increases.

Summary. We can see that our proposed RT-VNE achieves higher benefit revenue and acceptance ratio than state-of-the-art existing algorithms (Figures 6–8). This can be explained by the fact that RT-VNE accepts many more VNRs than related approaches. In addition, coordination of nodes and links mapping also leads to efficient mapping and consequently generates a higher income. We conclude that the effect of joint nodes and link mapping is significant. The proposed algorithm considers the neighborhood influence of substrate nodes, so with the limited embedding candidate number, it is highly chosen as efficient embedding candidates. Besides, the proposed algorithm checks the feasibility of embedding candidates before embedding, and it is necessary for efficient embedding.

Conclusion

VN embedding is a critical feature to NV technology. This article studied the VN embedding problem which is NP-hard and computationally intractable. We propose a new resource efficient algorithm for VNE which coordinates node and link mapping phases and handles the online dynamic VNR in the real world, resulting in the improvement of embedding efficiency. The incoming VNRs are grouped, analyzed, and ordered following selection rules. In each VNR, a VN node is selected as a root node and then mapped to root SN node into SN network. In comparison with some previous algorithms, the new strategies can markedly improve both resource utilization and long-term average revenue while the complexity is kept at an acceptable level.
Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIP) (No. 2016R1A5A1012966 and No. NRF-2015R1A2A1A01007400). Also, this work was partly supported by the Institute for Industrial Systems Innovation of Seoul National University.

ORCID iD
Duc-Lam Nguyen https://orcid.org/0000-0003-0161-3055

References
1. Bhardwaj S, Jain L and Jain S. Cloud computing: a study of infrastructure as a service (IAAS). Int J Eng Informa Tech 2010; 2(1): 60–63.
2. Fischer A, Botero JF, Beck MT, et al. Virtual Network Embedding: a survey. IEEE Comm Sur Tutor 2013; 15(4): 1888–1906.
3. Chowdhury NM and Boutaba R. Network virtualization: state of the art and research challenges. Comm Magazine 2009; 47(7): 20–26.
4. Bianchi F and Lo Presti F. A Markov reward based resource-latency aware heuristic for the Virtual Network Embedding problem. ACM SIGMETRICS Perform Eval Rev 2017; 44(4): 57–68.
5. Ricci R, Alfeld C and Lepreau J. A solver for the network testbed mapping problem. ACM SIGCOMM Comp Comm Rev 2003; 33(2): 65–81.
6. Zhu Y and Ammar MH. Algorithms for assigning substrate network resources to virtual network components. In: INFOCOM, Barcelona, 23–29 April 2006.
7. Chowdhury M, Rahman MR and Boutaba R. Vineyard: Virtual Network Embedding algorithms with coordinated node and link mapping. IEEE/ACM Trans Network 2012; 20(1): 206–219.
8. Andersen D. Theoretical approaches to node assignment (unpublished manuscript), https://www.cs.cmu.edu/~dga/papers/andersen-assign-abstract.html, 2002.
9. Yu M, Yi Y, Rexford J, et al. Rethinking Virtual Network Embedding: substrate support for path splitting and migration. ACM SIGCOMM Comp Comm Rev 2008; 38(2): 17–29.
10. Zhang Z, Su S, Zhang J, et al. Energy aware Virtual Network Embedding with dynamic demands: online and offline. Comp Networks 2015; 93: 448–459.
11. Chowdhury NM, Rahman MR and Boutaba R. Virtual Network Embedding with coordinated node and link mapping. In: INFOCOM, Rio de Janeiro, 19–25 April 2009, pp. 783–791. New York: IEEE.
12. Lisichka J and Karl H. A virtual network mapping algorithm based on subgraph isomorphism detection. In: Proceedings of the 1st ACM workshop on virtualized infrastructure systems and architectures, Barcelona, 16–21 August 2009, pp. 81–88. New York: ACM.
13. Cheng X, Su S, Zhang Z, et al. Virtual Network Embedding through topology-aware node ranking. ACM SIGCOMM Comp Comm Rev 2011; 41(2): 38–47.
14. Zhang S, Qian Z, Wu J, et al. Virtual Network Embedding with opportunistic resource sharing. IEEE Trans Parallel Dist Syst 2014; 25(3): 816–827.
15. Su S, Zhang Z, Liu AX, et al. Energy-aware Virtual Network Embedding. IEEE/ACM Trans Network 2014; 22(5): 1607–1620.
16. Heller B, et al. ElasticTree: saving energy in data center networks. In: NSDI’10 proceedings of the 7th USENIX conference on Networked systems design and implementation, San Jose, CA, 28–30 April 2010.
17. Beck MT, Fischer A, de Meer H, et al. A distributed, parallel, and generic Virtual Network Embedding framework. In: 2013 IEEE International Conference on Communications (ICC), New York, 9–13 July 2013, pp. 3471–3475. New York: IEEE.
18. Beck MT, Fischer A, Botero JF, et al. Distributed and scalable embedding of virtual networks. J Netw Comp Appl 2015; 56: 124–136.
19. Fajjari I, Saadi NA, Pujolle G, et al. VNE-AC: Virtual Network Embedding algorithm based on ant colony metaheuristic. In: 2011 IEEE international conference on communications (ICC), Kyoto, Japan, 5–9 June 2011, pp. 1–6. New York: IEEE.
20. Lu J and Turner J. Efficient mapping of virtual networks onto a shared substrate. Technical Report, Washington University in St. Louis, St. Louis, MI, January 2006.
21. Cheng X, Su S, Zhang Z, et al. Virtual Network Embedding through topology awareness and optimization. Comp Networks 2012; 56(6): 1797–1813.
22. Houidi I, Louati W and Zeghlache D. A distributed virtual network mapping algorithm. In: IEEE international conference on communications 2008 (ICC’08), Beijing, China, 19–23 May 2008.
23. Houidi I, Louati W, Zeghlache D, et al. Adaptive virtual network provisioning. In: Proceedings of the second ACM SIGCOMM workshop on virtualized infrastructure systems and architectures, New Delhi, India, 30 August–3 September 2010, pp. 41–48. New York: ACM.
24. Guo T, Wang N, Moessner K, et al. Shared backup network provision for Virtual Network Embedding. In: 2011 IEEE international conference on communications (ICC), Kyoto, Japan, 5–9 June 2011, pp. 1–5. New York: IEEE.
25. Duan J, Guo Z and Yang Y. Cost efficient and performance guaranteed Virtual Network Embedding in multicast fat-tree DCNs. In: 2015 IEEE conference on computer communications (INFOCOM), Kowloon, Hong Kong, 26 April–1 May 2015. New York: IEEE.
26. Nguyen LD, et al. RT-VNE: a real-time strategy for Virtual Network Embedding towards resource efficiency. In: 2017 international conference on information networking (ICOIN), Da Nang, Vietnam, 11–13 January 2017. New York: IEEE.

27. Zegura EW, Calvert KL and Bhattacharjee S. How to model an internetwork. In: Proceedings of the fifteenth annual joint conference of the IEEE computer societies (networking the next generation, INFOCOM ’96), vol. 2, San Francisco, CA, 24–28 March 1996, pp. 594-602. New York: IEEE.