Toward Profit-Seeking Virtual Network Embedding Algorithm via Global Resource Capacity

Long Gong*†, Yonggang Wen*, Zuqing Zhu† and Tony Lee‡
*School of Computer Engineering, Nanyang Technological University, Singapore 639798
Email: {n11036831e, ygwen}@ntu.edu.sg
†School of Information Science and Technology, University of Science and Technology of China, Anhui, PRC
Email: zqzhu@ieee.org
‡Department of Electronic Engineering, Shanghai Jiaotong University, Shanghai, PRC
Email: ttlee@sjtu.edu.cn

***Abstract*—In this paper, after proposing a novel metric, *i.e.*, global resource capacity (GRC), to quantify the embedding potential of each substrate node, we propose an efficient heuristic virtual network embedding (VNE) algorithm, called as *GRC-VNE*. The proposed algorithm aims to maximize the revenue and to minimize the cost of the infrastructure provider (InP). Based on GRC, the proposed algorithm applies a greedy load-balance manner to embed each virtual node sequentially, and then adopts the shortest path routing to embed each virtual link. Simulation results demonstrate that our proposed *GRC-VNE* algorithm achieves lower request blocking probability and higher revenue due to the more appropriate consideration of the resource distribution of the entire network, when compared to the two lastest VNE algorithms that also consider the resources of entire substrate network. Then, we introduce a classical reserved cloud revenue model, which consists of fixed revenue and variable one. Based on this revenue model, we design a novel admission control policy selectively accepting the VNR with high revenue-to-cost ratio to maximize the InP's profit based on an empirical threshold. Through extensive simulations, we observe that the optimal empirical threshold is proportional to the ratio of variable revenue to the fixed one.**

***Index Terms*—Network virtualization, Virtual network embedding (VNE), Global resource capacity (GRC), GRC-VNE, Reserved cloud revenue model, Admission policy**

I. INTRODUCTION

Recently, network virtualization has gained intensive attention from both research community [1–3] and industry [4, 5], as it provides a promising solution for future Internet [6] and works as an key enabler for cloud computing [7]. Network virtualization is to provision multiple virtual networks (VNs) over a substrate network for sharing computing and networking resources. A VN is a logical topology composed of a set of virtual nodes (*e.g.*, virtual routers) interconnected by corresponding virtual links over the substrate network. In this case, the conventional Internet service provider (ISP) can be decoupled into infrastructure providers (InPs) (*e.g.*, cloud providers) and service providers (SPs) (*e.g.*, cloud users). Several SPs dynamically construct VNs to aggregate the demands of the end-users by leasing infrastructures from the InPs. Given a set of VN requests (VNRs) with certain resources requirements on both nodes and links, the problem of finding a specific subset of nodes and links in the substrate network to satisfy each VNR is known as the virtual network embedding (VNE) problem [8]. In solving the VNE problem, InPs normally aim to maximize their revenue, with a underlied substrate network.

However, the VNE problem has been shown to be NP-hard [9]. Previous researches on the VNE problem [8, 10–16] relied on heuristic algorithms to balance the tradeoff between the performance and the computational complexity. Generally, some algorithms [10, 11] focused on the problem either in absence of node or link requirements, or with infinite resources on the nodes and links in the substrate network. Some others made efforts to either reduce the computational complexity by solving the node mapping and link mapping separately [13], or improve the performance by optimizing the two subproblems (*i.e.*, node mapping and link mapping) jointly [15, 17]. However, the former category relied on non-realistic assumptions that could not reveal the key point of the VNE problem, while for the latter one, it is difficult to obtain a good tradeoff between the performance and the computational complexity. Thus, recent works [8, 15, 16] proposed a new type of VNE algorithms, which we refer as coordinated VNE (CO-VNE). The CO-VNE algorithms aimed at achieving the best tradeoff between the performance and the computational complexity. More specifically, they solved the node mapping and link mapping separately, but taking link mapping constraints into consideration in the node mapping stage to mitigate the performance degradation due to the separated handling.

In this paper, we propose an efficient CO-VNE algorithm to maximize the revenue of the InP. We first formulate a novel metric, called as global resource capacity (GRC), to quantify the embedding potential of all the nodes in the substrate network. The calculation of GRC of each node takes the resources of the entire network into consideration. By expressing the GRC in a random walk form [18], we reveal the physical meaning behind it. Then, we design a novel CO-VNE algorithm based on GRC, which we refer as global resource capacity based virtual network embedding (*GRC-VNE*). This algorithm applies a load-balanced manner to embed the virtual node requires most (in terms of GRC) onto the node with the highest GRC in the substrate network, and leverages the shortest-path routing method for link mapping. Numerical

978-14799-3360-0/14/$31.00 ©2014 IEEE

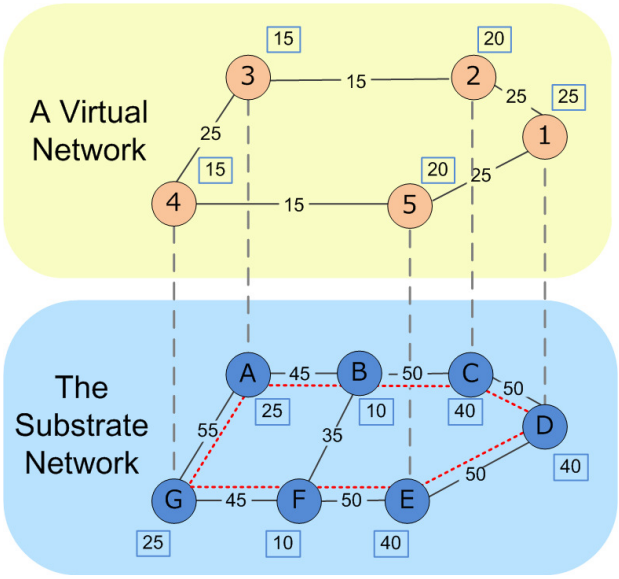


Fig. 1. An example of VNE process.

results demonstrate that the proposed algorithm outperforms two benchmark algorithms, which also consider the resources of the entire network. Then we introduce the classical reserved cloud revenue model, where revenue of a VNR consists of fixed and variable parts. Based on this revenue model, we develop a novel threshold based admission control policy that rejects VNR with a low revenue-to-cost ratio, to maximize the profit of the InP. Simulation results indicate that the optimal threshold can be obtained to maximize the revenue, which depends on the relative weight of the fixed and the variable revenue.

The rest of this paper is organized as follows. We formulate the VNE problem in Section II. The VNE characterization is described in Section III. The details of the proposed GRC-VNE algorithm are discussed in Section IV. Section V shows simulation setup and results for performance evaluation of the GRC-VNE algorithm. The optimization of GRC-VNE based on a reserved cloud revenue model is investigated in Section VI. Finally, Section VII summarizes the paper.

II. Problem Models and Formulation

In this section, we present a generic description of the VNE problem, including VNE models, VNE process, VNE revenue model, and VNE objective.

A. VNE Models

In a VNE problem, resources (computing resources for nodes and bandwidth resources for links) from a substrate network are allocated to meet the requirement in a VNR. When the InP gets a VNR, the InP should decide whether to accept it or not. If the VNR is accepted, then certain computing resources on corresponding substrate nodes and bandwidth on corresponding substrate links should be allocated to meet the demands of the VNR, and when the VNR departs, the allocated resources would be released. The models for both the substrate network and the VNR are detailed as follows.

1) Substrate Network: A substrate network (SN) can be modeled as an undirected graph, denoted as $G^s(V^s, E^s)$, where V^s is the set of substrate nodes and E^s is the set of substrate links. For each substrate node $v^s \in V^s$, it has a computing capacity (*e.g.*, CPU cycles) of c_{v^s}; for each substrate link $e^s \in E^s$, it has a bandwidth capacity of b_{e^s}. In this paper, we also adopt the notation of P_{G^s} as the set of paths in the substrate network. An example of a substrate network is illustrated in the bottom of Fig. 1. The numbers around the nodes and links are the available resources for them.

2) Virtual Network Request: A VNR, ϖ, can also be modeled as an undirected graph, denoted as $G^r(V^r, E^r)$, where V^r is the set of virtual nodes and E^r is the set of virtual links. For each virtual node $v^r \in V^r$, it is associated with a computing resource demand of c_{v^r}; for each virtual link $e^r \in E^r$, it is associated with a bandwidth demand of b_{e^r}. An example of a VNR is illustrated in the upper of Fig. 1. In this paper, we also assume that VNR arrival process is a Poisson process, with a time-invariant rate of λ. Each VNR is also associated with two time-domain parameters: t_ϖ for the arrival time of the VNR and a finite value τ_ϖ for the lifetime of the VNR.

B. VNE Process

The generic VNE process, as illustrated in Fig. 1, consists of two key components, *i.e.*, node mapping and link mapping.

1) Node Mapping: In the node-mapping stage, the InP tries to find, for each node of the VNR, a unique substrate node that has enough available computing resources to meet its resource demand. Mathematically, the node mapping can be described as a one-to-one mapping, *i.e.*, $F_N : V^r \mapsto V^s$, such that,

$$F_N(v^r) = v^s_r, \qquad v^r \in V^r, v^s_r \in V^s, \tag{1}$$

under the following two conditions, including,

- We have $F_N(v^{r,1}) = F_N(v^{r,2})$ for any $v^{r,1}, v^{r,2} \in V^r$ if and only if $v^{r,1} = v^{r,2}$;
- We have $c_{v^r} \leq c_{v^s_r}$.

For example, as shown in Fig. 1, the node mapping for the VNR is $\{1 \to D, 2 \to C, 3 \to A, 4 \to G, 5 \to E\}$ [1].

2) Link Mapping: In the link-mapping stage, for two adjacent nodes in the VNR, the InP finds a set of paths between the two mapped nodes in the substrate network, whose total available resources are larger than demand in the virtual link of the VNR. Two flavors of link mapping have been proposed previously, including,

- Multi-Path Mapping [8]: in this case, with path-splitting technique [12], a single virtual link can be mapped onto several different substrate paths.
- Single-Path Mapping [14]: in this case, a single virtual link can only be mapped onto one substrate path.

[1]Note that virtual nodes from different VNRs can be mapped onto the same substrate node as long as the available computing resources are sufficient.

In this work, we only consider the single-path mapping case; our results can easily be extended to the multi-path mapping case. Under the single-path mapping case, the link mapping can be represented by a mapping $F_L : E^r \mapsto P_{G^s}$, such that,

$$F_L(e^r) = p^s_{e^r}, \qquad e^r \in E^r, p^s_{e^r} \in P_{G_s}, \tag{2}$$

under the following capacity constraints, *i.e.*,

$$b_{e^r} \leq \min_{e^s \in p^s_{e^r}} b_{e^s}. \tag{3}$$

For example, as shown in Fig. 1, the link mapping for the VNR is $\{(1,2) \rightarrow (D,C), (2,3) \rightarrow (C,B,A), (3,4) \rightarrow (A,G), (4,5) \rightarrow (G,F,E), (5,1) \rightarrow (E,D)\}$.

C. VNE Revenue Model

For each VNR, we adopt a "pay-per-user" revenue model, based on the "on-demand" cloud service price scheme by Amazon Web Services (AWS) [19]. Specifically, for a VNR, ϖ, the revenue generated for the InP is given by

$$R(\varpi) = \begin{cases} R_0(\varpi) \cdot \tau_\varpi, & \text{if } \varpi \text{ is accepted} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where $R_0(\varpi)$ is the revenue per time-unit for ϖ., and τ_ϖ denotes the lifetime of the VNR. $R_0(\varpi)$ is defined as the summation of contributions from the computing resource demand and the bandwidth demand of ϖ, *i.e.*,

$$R_0(\varpi) = \alpha \cdot \sum_{v^r \in V^r} c_{v^r} + \beta \cdot \sum_{e^r \in E^r} b_{e^r}, \tag{5}$$

where α and β are the unit price charged for computing resources and bandwidth resources, respectively.

Notice that our revenue model captures the essential of cloud service revenue model, by considering both the resource demand and the usage duration of a VNR. On the other hand, existing revenue models, as defined in [8, 12, 15, 16], are too simple to capture the cloud service model, adopted on the market. Specifically, all these papers used the same revenue model, which is totally determined by the resource demands of the VNR (*i.e.*, $R_0(\varpi)$). They do not consider the lifetime of the VNR.

D. VNE Objective

Similar to the previous work [20], the major interest of this paper is to maximize the long-term time-average revenue [2] of the InP, which is defined as,

$$R_{G^s} = \lim_{T \to \infty} \frac{\sum_{\varpi \in \Omega_T} R(\varpi)}{T}, \tag{6}$$

where $\Omega_T = \{\varpi | 0 \leq t_\varpi \leq T\}$ denotes the set of VNRs arriving before time instance T.

Obviously, the revenue for an InP is related to two additional metrics, *i.e.*,

- VNR Blocking Probability: it is defined as

$$p_b = \lim_{T \to \infty} \frac{\phi_b(T)}{\phi(T)}, \tag{7}$$

where, $\phi_b(T)$ and $\phi(T)$ are the numbers of blocked (or rejected) VNRs and total VNRs in time period T, respectively.

- VNR Revenue-to-Cost Ratio: it is defined as

$$\Im(\varpi) = \frac{R_0(\varpi)}{C_0(\varpi)}, \tag{8}$$

where, $C_0(\varpi)$ is the resource cost per time-unit to serve a VNR, ϖ, similar to previous work [12, 15], $C_0(\varpi)$ is given by,

$$C_0(\varpi) = \sum_{v^r \in V^r} c_{v^r} + \sum_{e^r \in E^r} \mid F_L(e^r) \mid b_{e^r} \tag{9}$$

where, $\mid F_L(e^r) \mid$ is the length of path $F_L(e^r)$ in terms of hops.

III. VNE Characterization

In this section, we first establish an upper bound for the revenue and illustrate two design guidelines for VNE algorithms.

A. Revenue Upper Bound

For a given substrate network $G^s(V^s, E^s)$, we can establish an upper bound of revenue, given by

$$R_{G^s} \leq \alpha \cdot \sum_{v^s \in V^s} c_{v^s} + \beta \cdot \sum_{e^s \in E^s} b_{e^s} \triangleq R^u_{G^s}. \tag{10}$$

This bound can be established via the following hypothetical situation,

- Each VNR is an equivalent to the substrate network;
- Subsequent VNR arrives at the instance when previous VNR expires.

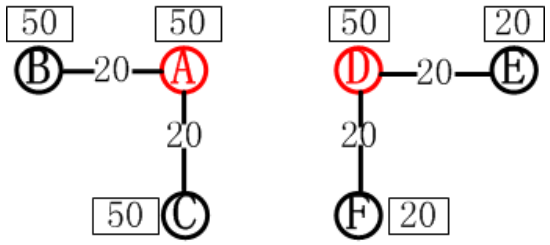
In this situation, each VNR is satisfied with the minimum amount of substrate resources and the substrate network is fully occupied. As such, the time-accumulate revenue would maximize, in turn, the revenue would maximize.

B. Revenue-Penalty Factors

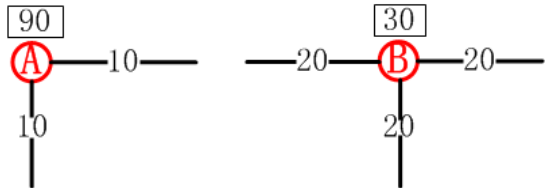
In reality, the observed revenue of the InP is much less than the upper bound, rendered by the underlying substrate network. This gap results from the irregularity associated with the VNR arrivals, deviating from the aforementioned two ideal conditions. With a set of non-ideal VNRs, the following two cases would add penalty to the revenue by the InP, including,

1) *Case 1*: a VNR is rejected because the substrate network cannot accommodate it due to insufficient resources. The resource deficiency could be resulted from the fundamental resource limitation or resource wasted by inappropriate VNE algorithms.
2) *Case 2*: even if a VNR is accepted, a subset of virtual edges are mapped onto paths in the substrate network that span more than one edge.

[2]Unless stating explicitly, we use "revenue" to stand for "long-term time-average revenue" in the following parts of this paper



(a) The locality problem



(b) The "big island" problem

Fig. 2. An example of disadvantages of LRC

Understanding of these penalty factors suggests some useful guidelines for us to design effective VNE algorithms. For *Case 1*, we could minimize the blocking probability to make the substrate network accept the most VNRs with its limited resources. For *Case 2*, we could maximize the revenue-to-cost ratio for each VNR to cost as less as possible resources on a single VNR. In next section, we will propose our VNE algorithm, by adopting these heuristics.

IV. VNE Algorithm Based on Global Resource Capacity

It is known that the VNE problem is NP-hard [9]. Consequently, research efforts have been focused on heuristic algorithms to achieve a reasonable trade-off between the computational complexity and the VNE design performance. In this section, we propose a VNE algorithm in the same family. More specifically, the proposed algorithm can be categorized as a CO-VNE algorithm. We first formulate a proper metric, which is called as global resource capacity (GRC), to quantify the embedding potential of each node in the substrate network. Then, we conduct a greedy node mapping based on GRC followed by a shortest-path based link mapping.

A. Global Resource Capacity

1) Motivations: As one of the earliest attempts to consider link mapping constraints in node mapping stage, Yu *et al.* [12] formulated local resource capacity (LRC), which was defined as the product of a node's computing resources and the summation of bandwidth resources of its incident links. However, LRC has the following disadvantages,

- *the locality problem*: as explained in [15], the LRC of a node only considers the resources of the node itself together with its incident links, which cannot reveal the real embedding potential of the node. For example, as shown in Fig. 2(a), nodes A and D have the same LRC values as $50\times(20+20)=2000$ units. While the available resources of the adjacent nodes of node A are more than those of the adjacent nodes of node D, the embedding potential of node A should be larger than that of node D.
- *the "big island" problem*: when the load distribution in a network gets imbalanced, the nodes that still have large amount of available resources will become "big islands" in the network and they cannot be utilized properly based on LRC. For example, as shown in Fig. 2(b), node A still has a fairly large amount of computing resources, and hence even when its incident links' bandwidth resources are very limited, its LRC value is still the same as that of node B.

Even though the locality problem can be overcome by using additional topology-aware metrics [15, 16], the "big island" problem has yet been addressed properly.

2) Definition: In order to overcome the locality problem and the "big island" problem of LRC, we formulate the metric of global resource capacity (GRC), to quantify the embedding potential of each node in the substrate network. Specifically, given a network topology $G(V,E)$, the computing resource $c_v, v\in V$, and the bandwidth resource $b_{(u,v)}, (u,v)\in E$, the GRC $r(u)$ of node $u\in V$ is formulated as

$$r(u)=(1-d)\cdot\hat{c}_u+d\sum_{v\in N(u)}\frac{b_{(u,v)}}{\sum_{x\in N(v)}b_{(x,v)}}\cdot r(v) \quad (11)$$

where d is a constant damping factor within $(0,1)$, $N(u)$ indicates the set of adjacent nodes of node u, and $\hat{c}_u$ is the normalized computing resources on node u as

$$\hat{c}_u=\frac{c_u}{\sum_{v\in V}c_v},\quad \forall u\in V.$$

Using a vector format, we can calculate the GRCs for all the nodes as

$$\mathbf{r}=(1-d)\mathbf{c}+d\mathbf{M}\mathbf{r}, \quad (12)$$

where $\mathbf{r}=(r(1),r(2),...,r(|V|))^T$, $\mathbf{c}=(\hat{c}_1,\hat{c}_2,...,\hat{c}_{|V|})^T$, and $\mathbf{M}$ is a transition matrix with the dimension as $|V|\times|V|$. Each entry in $\mathbf{M}$ is defined as

$$m_{(u,v)}=\begin{cases}\frac{b_{(u,v)}}{\sum_{x\in N(v)},b_{(x,v)}} & (u,v)\in E\\ 0, & \text{otherwise}\end{cases}. \quad (13)$$

We first need to establish the uniqueness of our GRC definition.

Proposition 4.1: Matrix $\mathbf{I}-d\mathbf{M}$ is reversible, and hence a unique vector $\mathbf{r}$ can be obtained from Eq. (12).

Proof: With Eq. (13), we have

$$\sum_{u\in V}m_{(u,v)}=\sum_{u\in N(v)}\frac{b_{(u,v)}}{\sum_{x\in N(v)}b_{(x,v)}}=1.$$

According to the Gershgorin Circle Theorem [21], we conclude that $\|\mathbf{M}\|\leq 1$. If the matrix $\mathbf{I}-d\mathbf{M}$ is irreversible, there must be a non-zero vector $\mathbf{x}$ that can satisfy $(\mathbf{I}-d\mathbf{M})\mathbf{x}=0$. It follows that $d\mathbf{M}\mathbf{x}=\mathbf{x}$, and $||\mathbf{x}||=||d\mathbf{M}\mathbf{x}||\leq d\|\mathbf{M}\|\|\mathbf{x}\|$. As $\mathbf{x}$ is a non-zero vector, we can conclude that $\|\mathbf{M}\|\geq\frac{1}{d}>1$.

However, this conclusion is contradicted with $\|\mathbf{M}\| \leq 1$. Therefore, we prove that $\mathbf{I} - d\mathbf{M}$ is reversible. ■

It follows that the unique solution of Eq. (12) is given by

$$\mathbf{r} = (1-d)(\mathbf{I} - d\mathbf{M})^{-1}\mathbf{c}. \tag{14}$$

3) Physical Meaning:

Proposition 4.2: Eq. (11) can be rewritten in a random walk form [18] as

$$r(u) = \sum_{v \in V} \hat{c}_v \left((1-d)\delta_{u,v} + \sum_{p \in P_{v,u}} B(p)(1-d)d^{H(p)} \right), \tag{15}$$

where $\delta_{u,v}$ is the Kronecker's delta, defined as $\delta_{u,v} = \begin{cases} 1 & \text{if } u = v \\ 0 & \text{if } u \neq v \end{cases}$, $P_{v,u}$ denotes the set of all the paths from node v to node u, $B(p) = \frac{\prod_{k=1}^{n-1} b_{(v_k, v_{k+1})}}{\prod_{k=1}^{n-1} \sum_{x \in N(v_k)} b_{(x, v_k)}}$ for any path $p = (v_1, v_2, ..., v_n) \in P_{v_1, v_n}$, and $H(p)$ is a function that returns the hops of path p.

Proof: Substitute Eq. (15) into the right-hand side of Eq. (11), we obtain

$$\begin{aligned}
&(1-d)\cdot\hat{c}_u + d\sum_{v\in N(u)} \frac{b_{(u,v)}}{\sum_{w\in N(v)} b_{(w,v)}} \cdot r(v) \\
&= (1-d)\cdot\hat{c}_u + d\sum_{v\in N(u)} \frac{b_{(u,v)}}{\sum_{w\in N(v)} b_{(w,v)}} \\
&\quad \cdot \sum_{x\in V} \hat{c}_x \left((1-d)\delta_{v,x} + \sum_{p\in P_{x,v}} B(p)(1-d)d^{H(p)} \right) \\
&= \sum_{x\in V} \hat{c}_x \left((1-d)\delta_{u,x} + \sum_{p\in P_{x,u}} \delta_{p,(x,u)} B(p)(1-d)d \right. \\
&\quad \left. + \sum_{p\in P_{x,v}} (1-d)d^{H(p)+1} \sum_{v\in N(u)} B(p) \frac{b_{(u,v)}}{\sum_{w\in N(v)} b_{(w,v)}} \right) \\
&= \sum_{x\in V} \hat{c}_x \left((1-d)\delta_{u,x} + \sum_{p\in P_{x,u}} B(p)(1-d)d^{H(p)} \right) \\
&= r(u)
\end{aligned}$$

■

Eq. (15) indicates that the GRC $r(u)$ of any node $u \in V$ is a weighted summation of the resources of the entire network, *i.e.*, other nodes in the network also contribute to the GRC of node u. This is the reason why we refer $r(u)$ as the "global" resource capacity of node u. Specifically, the more computing resources a node $v, v \neq u$ has, the shorter the paths $p \in P_{v,u}$ are, and the more bandwidth the paths $p \in P_{v,u}$ have, the more contributions node v can make to the GRC of node u.

4) Calculation: The complexity of calculating the GRC vector with Eq. (14) directly is $O(|V|^3)$. In this work, we adopt a simple iterative algorithm to calculate the GRC vector more efficiently

$$\mathbf{r}_{k+1} = (1-d)\mathbf{c} + d\mathbf{M}\mathbf{r}_k, \tag{16}$$

where $\mathbf{r}_k$ is the GRC vector after k iterations. We terminate the iterations when the improvement between two adjacent iterations is smaller than a pre-set threshold. It can be shown that this algorithm always converges to a stationary solution, as it is equivalent to the Jacobi algorithm [21] for solving linear equations. *Algorithm 1* shows the details of this algorithm.

Algorithm 1: Calculation of GRC vector

input : Network topology $G(V, E)$, pre-set small positive threshold σ
output: GRC vector $\mathbf{r}$

```
calculate matrix M and vector c respectively;
r_0 = c;
k = 0;
Δ = ∞;
while Δ ≥ σ do
    r_{k+1} = (1 − d)c + dMr_k;
    Δ = ‖r_{k+1} − r_k‖;
    k = k + 1;
end
r = r_k;
```

Since in each iteration, the number of operations is proportional to $O(|V|^2)$ and in the practical case, $\sigma \ll 1$, the number of iterations is proportional to $\log \frac{1}{\sigma}$, the complexity of *Algorithm* 1 is $O(|V|^2 \log \frac{1}{\sigma})$ [22].

B. Greedy Node Mapping

In the node-mapping stage, we adopt a greedy mapping algorithm. It works as follows. After calculating the GRC vector for both the substrate network and the VNR, we sort the nodes of the two topologies in a descending order according to the GRCs, respectively. From the perspective of load balance of the substrate network, the greedy node mapping is conducted by processing the two node lists from top to down and mapping nodes one by one, similar to the *merge-sort* algorithm, *i.e.*, as long as the computing requirement can be satisfied, a virtual node with the largest GRC in the VNR is always mapped onto the substrate one that also has the largest GRC. If the computing resource requirement cannot be satisfied by using any of the substrate nodes, the VNR is blocked.

Algorithm 2 shows the details of the greedy node mapping algorithm. The time complexity of this algorithm is $O(|V^s||V^r|)$.

C. Shortest-Path based Link Mapping

In the link-mapping stage, we adopt the shortest-path based link mapping algorithm, which aims to minimize the cost. Specifically, for each edge in the VN request, we use the Dijkstra's algorithm to calculate the shortest path between the corresponding nodes in the substrate network. This algorithm

Algorithm 2: Greedy Node Mapping

input : VNR ϖ, $G^r(V^r, E^r)$, substrate network $G^s(V^s, E^s)$, the GRC vector $\mathbf{r}^s$ for substrate network, and the GRC vector $\mathbf{r}^r$ for ϖ
output: Node mapping F_N

$F_N \leftarrow \mathbf{0}$;
$Count = 0$;
sort $\mathbf{r}^s$ in a descending order to get $\mathbf{r}^s_{sort}$;
sort $\mathbf{r}^r$ in a descending order to get $\mathbf{r}^r_{sort}$;
for *each virtual node* v^r *in the order of* $\mathbf{r}^r_{sort}$ **do**
  for *each unselected substrate node* v^s *in the order of* $\mathbf{r}^s_{sort}$ **do**
    if $c_{v^s} \geq c_{v^r}$ **then**
      $F_N(v^r) = v^s$;
      $Count = Count + 1$;
      break;
    end
  end
end
if $Count = \mu$ **then**
  continue with link mapping;
else
  mark ϖ as blocked;
  wait for the next VNR;
end

Algorithm 3: Shortest-Path based Link Mapping

input : VNR ϖ, $G^r(V^r, E^r)$, substrate network $G^s(V^s, E^s)$, node mapping F_N
output: Link mapping F_L

$F_L \leftarrow \mathbf{0}$;
for *each virtual link* $e^r = (v^{r,1}, v^{r,2})$ *in* ϖ **do**
  $G^s_{temp} \leftarrow G^s$;
  for *each substrate link* e^s *in* G^s_{temp} **do**
    if $b_{e^s} < b_{e^r}$ **then**
      cut e^s in G^s_{temp};
    end
  end
  try to find a shortest path $p^s_{e^r}$ from $F_N(v^{r,1})$ to $F_N(v^{r,2})$ in G^s_{temp};
  if *cannot find a path* **then**
    mark ϖ as blocked;
    wait for the next VNR;
  else
    update bandwidth of E^s in G^s;
    $F_L(e^r) \leftarrow p^s_{e^r}$;
  end
end

is made more efficient by pre-cutting all the links in the substrate that do not have enough available bandwidth resources. If the link mapping fails (*i.e.*, any virtual link can not be embedded successfully), we restore the substrate network status and mark the VNR as blocked.

The details of the link mapping is shown in *Algorithm 3*. The time complexity of this algorithm is $O(|E^r||E^s|log|V^s|)$.

Summing all these three sub-algorithms, we determine that the time complexity of our *GRC-VNE* algorithm is $O((|V^s|^2 + |V^r|^2)\log(\frac{1}{\sigma}) + |V^s||V^r| + |E^r||E^s|log|V^s|)$.

V. Performance Evaluation

In this section, we perform numerical simulations to compare the performance of our GRC-VNE with two latest VNE algorithms, *i.e.*, the RW-MM-SP algorithm [15] and the TA algorithm [16], which also consider the resources of the entire network when doing node mapping. In this study, we consider two performance metrics, including the revenue of the Eq. (6) and the VNR blocking probability of Eq. (7).

A. Simulation Model

In this research, we adopt the similar simulation model as those of previous work [12, 15]. The topologies of the substrate network and the VNRs are randomly generated by the GT-ITM tool [23]. For the substrate network, the initial available computing resources and bandwidth resources are randomly selected by uniform distribution. In each VNR, the computing resource demand of each virtual node and the bandwidth requirement of each virtual link are also randomly selected. The number of virtual nodes in a VNR is selected from 2 to 20 according to the uniform distribution. The virtual link connectivity rate, which is the probability of any two nodes are connected in a VNR, is set to η. Thus, the average number of links in a VNR is $\frac{n(n-1)}{2} \cdot \eta$, where n is the number of virtual nodes. As assumed in Subsection II-A, the VNRs arrive one by one, formulating a Poisson process with an average arrival rate of λ requests per time unit, and the lifetime of each request follows the negative exponential distribution with an average of $\frac{1}{\mu} = 500$ time units. Hence, the load of the VNRs can be quantified with $\lambda \cdot \frac{1}{\mu}$ in Erlangs.

TABLE I
Simulation Parameters

Node number of the substrate network	100
Link number of the substrate network	570
Minimum degree of the substrate topology	4
Maximum degree of the substrate topology	20
Average degree of the substrate topology	11.4
Initial available computing resources on substrate nodes	50-100 units
Initial available bandwidth resources on substrate links	50-100 units
Average lifetime of the VNRs	500 time-units
Bandwidth demand of a virtual link	0-50 units
Computing resource demand of a virtual node	0-50 units
Node number in a VNR	2-20

Table I summarizes the parameters for our simulations.

B. Benchmark Performance Comparisons

We first evaluate the performance of the aforementioned three VNE algorithms using a fixed traffic load as 25 Erlangs and a fixed link connectivity rate as 0.5 for the VNRs, and run the simulations for 50,000 time units to see their operations in a long run. In the simulations, we set: $\alpha = \beta = 1$, $\sigma =$

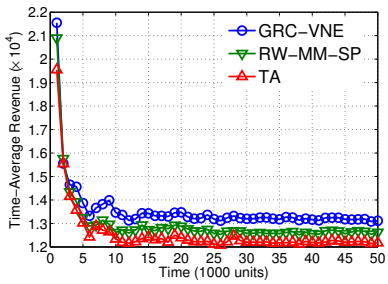


Fig. 3. Comparisons of time-average revenue in a long run

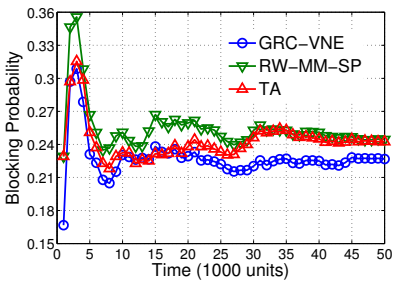


Fig. 4. Comparisons of blocking probability in a long run

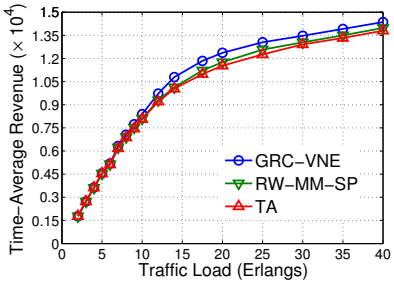


Fig. 5. Comparisons of time-average revenue under different traffic loads in stationary state

0.00001, and $d = 0.85$. All the parameters of RW-MM-SP and TA are from [15] and [16], respectively.

In Fig. 3, we compare the revenue from all three algorithms. It can been seen that our GRC-VNE outperforms the other two algorithms. The revenue advantage of our algorithm can be traced back to the fact that the GRC could quantify the embedding potential more efficiently by overcoming both the locality problem and the "big island" problem of the LRC. As a result, under the offered load, our algorithm will experience a smaller blocking probability for the VNR. Under our assumed stochastic traffic model, the revenue can be approximated by

$$\bar{R}_T \simeq (1 - p_b)\lambda \bar{R}_0(\varpi)\bar{\tau}_\varpi, \tag{17}$$

where $\bar{R}_0(\varpi)$ and $\bar{\tau}_\varpi$ refer to the average revenue per request and the average lifetime per request. This observation can be verified in Fig. 4, where the blocking probabilities of all the algorithms are compared. Notice that our algorithm has the lowest long-term blocking probability. As such, we expect that our algorithm would generate the highest revenue.

C. Sensitivity Analysis

In this subsection, we compare the performance of the three algorithms, under different traffic loads and link connectivity rates for VNRs.

First, in Fig. 5 and 6, we plot the revenue and the blocking probability, for the three algorithms, under different traffic loads. It can be seen that the GRC-VNE algorithm achieves the highest revenue for all traffic loads. It can also be seen that the revenue increases as the traffic load increases, in a concave manner. It can be understood as follows. As the traffic load increases, the substrate network performs closer to its capacity. As a result, additional VNRs might consume more resources if accepted, resulting in a lower revenue-to-cost ratio. Therefore, the revenue will flatten out as the traffic load increases further. This rationale is further verified by the observation of the blocking probability for all three algorithms will merge as the traffic load increases.

Second, the same sensitivity analysis is performed by varying the link connectivity rate for the VNRs. The results are plotted in Fig. 7 and 8. It can be seen that our algorithm generates the highest revenue and the lowest blocking probability. Moreover, we notice that the revenue first increases and then decreases as the link connectivity rate rises, resulting in a peak revenue when the link connectivity rate is a fixed value. This observation is rendered by a fundamental trade-off between the blocking probability and the revenue per request, as shown in Eq. (17). As the link connectivity rate rises, the blocking probability increases monotonically; while the average revenue per request also increases as it requires more resources. Therefore, at one point, the marginal penalty due to the blocking probability increasing is fully compensated by the marginal gain of increased revenue per request.

TABLE II
AVERAGE RUNNING TIME TO EMBED A SINGLE VNR (UNIT: SECONDS)

Link Connectivity Rates	RW-MM-SP	TA	GRC-VNE
0.2	1.036	1.072	0.447
0.4	4.302	3.688	0.607
0.6	6.201	5.467	1.062
0.8	6.871	6.611	1.304

D. Time Complexity

In this subsection, we compare the average running time for each algorithm to embed a VNR on the same substrate network. This metric can be used as an indicator for the time complexities of all the three algorithms. Our simulation environment is Matlab R2012a running on a computer with 3.10 GHz Intel Core i3-2100 CPU and 2.00 GB RAM. The average running times for the three algorithms under different link connectivity rates are compared in Table. II.

We observe that our algorithm has the lowest running time for all the cases. In general, our running time is approximately 20-30% of the running time for other two algorithms. Part of this running time gain originates from an optimized step in our algorithm, *i.e.*, we pre-cut all infeasible links before embedding virtual links, and only need to execute the shortest path algorithm once for each virtual link. While, for the K-shortest-path scheme in the other two VNE algorithms, the infeasible links are still considered in path computation and time is wasted on them.

VI. VNE Algorithm Optimization under Cloud Revenue Model

In this section, we apply the proposed *GRC-VNE* algorithm under a cloud computing environment, which adopts a revenue scheme similar to the reserved instance in AWS [19]. In this

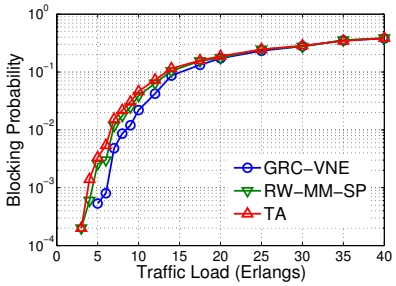


Fig. 6. Comparisons of blocking probability under different arrival rates in stationary state

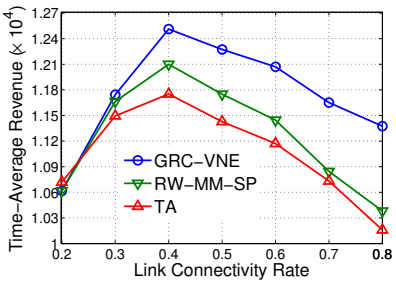


Fig. 7. Comparisons of time-average revenue under different link connectivity rates in stationary state

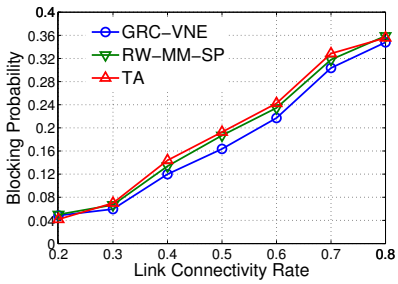


Fig. 8. Comparisons of blocking probability under different link connectivity rates in stationary state

practical situation, we notice that the *GRC-VNE* algorithm can be optimized for revenue, by adopting an empirical revenue-to-cost-ratio threshold based admission control policy.

A. Reserved Cloud Revenue Model

In practice, cloud providers often adopt a reserved price model. Specifically, the cloud user signs a long-term contract (*e.g.*, service agreement level) with the cloud provider. In this contract, the user would pay the cloud provider a fixed amount in the beginning and make additional payment for actual usage. The additional payment is usually a "pay-per-use" model, whose corresponding revenue model is assumed in Subsection II-C. This revenue model is corresponding to "reserved instance" in cloud price model.

Under this revenue model, for each VNR, its revenue consists of two parts, including:

- *Fixed Revenue Part*: this part of the revenue corresponds to the amortized one from the lump-sum payment of the user to the cloud provider, over the contract period.
- *Variable Revenue Part*: this part of the revenue corresponds to the usage payment from serving one specific VNR.

In this model, for a VNR ϖ, the total revenue obtained from it can be derived as

$$\hat{R}(\varpi) = R_f + R_v, \tag{18}$$

where the variable revenue part is defined in Eq. (4). The fixed revenue part is $R_f = \frac{P_T}{n_T}$, where P_T is the lump-sum payment for duration T and n_T is the total number of VNRs in that duration. In our model, the VNR arrives as a stochastic process; as a result, n_T is a random variable. Therefore, without loss of generality, we assume that the fixed part of revenue is random.

B. Admission Control Policy based on Revenue-to-Cost-Ratio Threshold

Our proposed *GRC-VNE* suffers from its greedy nature. Specifically, the *GRC-VNE* accepts a VNR as long as the request can be served by its substrate network. This admission control policy could penalize the revenue for the cloud providers, because it is quite possible to accept a VNR that would consume a large amount of resources. This would penalize future VNRs that would consume less resources but offer decent revenues. Therefore, a better request admission control policy should be in order, so as to increase the profit.

In this work, we propose a novel admission control policy, based on the revenue-to-cost-ratio threshold. With this policy, the cloud provider would accept a VNR only if the hypothetical revenue-to-cost ratio, $\Im(\varpi)$ (*i.e.*, $\frac{\hat{R}(\varpi)/\tau_\varpi}{C_0(\varpi)}$), is above a prefixed empirical threshold Ξ.

We use numerical simulations to investigate the impacts of the revenue-to-cost-ratio threshold Ξ and the fixed revenue part R_f to the performance of the revenue-driven *GRC-VNE* algorithm (*i.e.*, the *GRC-VNE* algorithm with the threshold based admission control policy). The variable revenue part R_v of a VNR is set to be equal to its total resource requirement times its lifetime, and the fixed revenue part R_f is selected from U_{min} to U_{max} according to an uniform distribution, denoted as $U(U_{min}, U_{max})$. The other parameters are the same as those in Subsection V-B. In Fig. 9, we plot the revenue in a long run for three different thresholds, in which R_f is taken randomly from the uniform distribution of $U(20K, 1M)$. Notice that, the revenue varies for different thresholds. This effect will be treated in more details, in next subsection.

C. Optimizing Revenue-to-Cost-Ratio Threshold

In this subsection, we will show that the revenue can be maximized by an optimal threshold and investigate how the optimal threshold vary as the system parameters change.

In Fig. 10, we plot the normalized revenue as a function of the threshold Ξ, for three different R_f distributions. The normalized revenue is defined as the revenue normalized by that of GRC-VNE with the reserved revenue model but no revenue-to-cost-ratio threshold based admission control policy. It can be seen that, when threshold Ξ increases, the revenue tends to first increase and then decrease. This observation can be understood as the trade-off between the blocking probability and the revenue intake. As the threshold Ξ increases, the blocking probability would increases, while the average revenue per request also increases. When the marginal change of these parameters match, there will be a stationary point for which the revenue is maximized. Therefore, we conclude that a unique optimal empirical threshold exists for maximizing the revenue.

We further investigate the relationship between the optimal threshold and the system parameters. Specifically, we believe

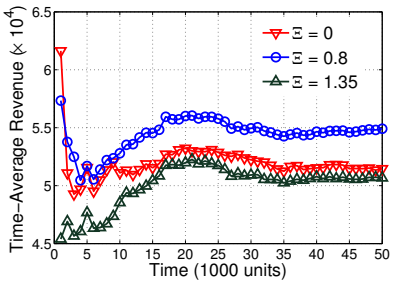


Fig. 9. Comparisons of time-average revenue in a long run under different thresholds

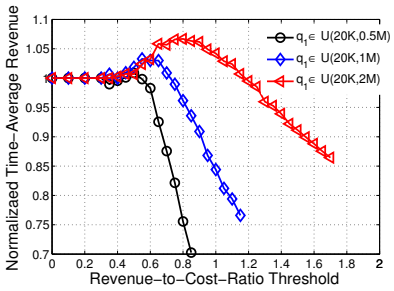


Fig. 10. The relation between the threshold and time-average revenue under different fixed revenues

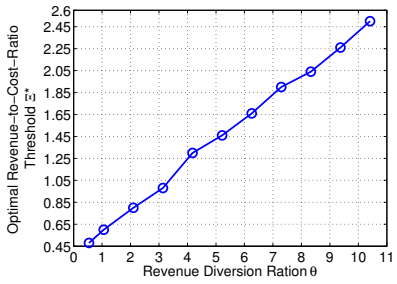


Fig. 11. The relation between the optimal threshold and the revenue diversion ration

that the optimal threshold is highly correlated with the ratio between the fixed revenue part and the variable revenue part, *i.e.*, $\theta = \frac{E\{R_f\}}{E\{R_v\}}$. In Fig. 11, we plot the optimal threshold as a function of the revenue diversion ration (*i.e.*, the ratio between the fixed revenue part and the variable revenue part θ). We observe that the optimal threshold Ξ^* increases monotonically and linearly as the revenue diversion ration (θ) increases. This suggests that the relative weight of the two revenue components dictates the optimal threshold, providing a practical guideline to choose an appropriate empirical threshold for the cloud provider to maximize its profit.

VII. Conclusion

In this paper, we first introduced a novel metric, *i.e.*, GRC, to quantify the embedding potential of the node in the substrate network. Based on this new metric, we proposed a novel VNE algorithm. The algorithm used GRC to map virtual nodes onto substrate nodes in a load-balanced manner, followed by a shortest-path based link mapping. Our simulation results suggested that our algorithm outperformed two other VNE algorithms that also considered the resources of the entire network for node mapping. Moreover, under a practical cloud service model (*i.e.*, AWS reserved price model), we optimized the *GRC-VNE* algorithm by introducing a novel admission control policy, which accepts a VNR when its revenue-to-cost ratio is higher than a fixed empirical threshold. Numerical evaluations suggested that the revenue can be maximized by a well-chosen empirical threshold and the optimal empirical threshold depends on the two components of the revenue model.

Acknowledgments

This work was supported in part by the Program for New Century Excellent Talents in University (NCET) under Project NCET-11-0884, the National Natural Science Foundation of China (NSFC) under Project 61371117, the Fundamental Research Funds for the Central Universities (WK2100060010), and the Strategic Priority Research Program of the Chinese Academy of Sciences (XDA06010301).

References

[1] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *IEEE Computer*, vol. 38, pp. 34–41, Apr. 2005.
[2] J. Turner and D. Taylor, "Diversifying the Internet," in *Proc. IEEE GLOBECOM 2005*, pp. 754–760, Dec. 2005.
[3] N. Feamster, L. Gao, and J. Rexford, "How to lease the Internet in your spare time," *SIGCOMM Comput. Commun. Rev.*, vol. 37, pp. 61–64, Jan. 2007.
[4] Nicira. Its time to virtualize the network. [Online]. Available: http://nicira.com/sites/default/files/docs/It%20is%20Time%20To%20Virtualize%20the%20Network%20White%20Paper_2.pdf
[5] Cisco. Network virtualization: The new building blocks of network design. [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns340/ns517/ns431/ns725/net_implementation_white_paper0900aecd80707cb6.pdf
[6] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Netw.*, vol. 54, no. 5, pp. 862–876, Apr. 2010.
[7] F. Baroncelli, B. Martini, and P. Castoldi, "Network virtualization for cloud computing," *annals of telecommunications - annales des télécommunications*, vol. 65, pp. 713–721, 2010.
[8] N. Chowdhury, M. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. IEEE INFOCOM 2009*, pp. 783–791, Apr. 2009.
[9] D. Andersen, "Theoretical approaches to node assignment," Dec. 2002, unpublished Manuscript.
[10] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proc. IEEE INFOCOM 2006*, pp. 1–12, Apr. 2006.
[11] J. Lu and J. Turner, "Efficient mapping of virtual networks onto a shared substrate," Washington University in St. Louis, Tech. Rep., 2006.
[12] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 19–29, Apr. 2008.
[13] A. Razzaq and M. Rathore, "An approach towards resource efficient virtual network embedding," in *Proc. INTERNET 2010*, pp. 68–73, Sept. 2010.
[14] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "VNE-AC: virtual network embedding algorithm based on ant colony metaheuristic," in *Proc. IEEE ICC 2011*, pp. 1–6, Jun. 2011.
[15] X. Cheng *et al.*, "Virtual network embedding through topology-aware node ranking," *SIGCOMM Comput. Commun. Rev.*, vol. 41, pp. 39–47, Apr. 2011.
[16] S. Zhang, Z. Qian, J. Wu, and S. Lu, "An opportunistic resource sharing and topology-aware mapping framework for virtual networks," in *Proc. IEEE INFOCOM 2012*, pp. 2408–2416, Mar. 2012.
[17] W. Liu, Y. Xiang, S. Ma, and X. Tang, "Completing virtual network embedding all in one mathematical programming," in *Proc. ICECC 2011*, pp. 183–185, Sept. 2011.
[18] M. Brinkmeier, "PageRank revisited," *ACM Trans. Internet Technol.*, vol. 6, no. 3, pp. 282–301, Aug. 2006.
[19] [Online]. Available: http://aws.amazon.com/ec2/#pricing
[20] M. Chowdhury and M. Rahman, "ViNEYard : Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, 2012.
[21] H. Gene and F. Charles, *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)(3rd Edition)*, 3rd ed. Johns Hopkins University Press, Oct. 1996.
[22] M. Bianchini, M. Gori, and F. Scarselli, "Inside PageRank," *ACM Trans. Internet Technol.*, vol. 5, no. 1, pp. 92–128, 2005.
[23] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. IEEE INFOCOM 1996*, pp. 594–602, Mar. 1996.