Revenue-Driven Virtual Network Embedding Based on Global Resource Information

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, Hefei, China
Email: {zqzhu}@ieee.org
‡Department of Electronic Engineering, Shanghai Jiaotong University, Shanghai, China
Email: {ttlee}@sjtu.edu.cn

***Abstract*—Virtual network embedding (VNE), working as a key step for network virtualization, has recently gained intensive attentions from the research community. In this paper, we propose a novel VNE algorithm that aims to maximize the infrastructure provider's revenue from serving virtual network (VN) requests, with the help of the global resource information. The proposed algorithm, named as revenue-driven VNE (*RD-VNE*), adopts a node-ranking approach that takes the global resource information into account in a recursive manner to assist the greedy node mapping, and leverages the shortest-path routing for link mapping. Our simulation results suggest that the proposed VNE algorithm outperforms two existing VNE algorithms that also take global resource information into consideration, in terms of request blocking probability, and brings higher time-average revenue to the infrastructure provider (InP).**

***Index Terms*—Network Virtualization, Virtual Network Embedding (VNE), PageRank, Node Ranking, Global Resource Information, RD-VNE**

I. INTRODUCTION

Recently, Internet has been growing exponentially with larger number of nodes and end-users, higher volume of traffic, and more varieties of applications. However, owing to the competing tussle among different stakeholders, Internet was brought into "ossification", which impeded the developments of new networking solutions and architectures greatly [1–3]. To this end, people rely on the network virtualization [4–6] to change how Internet is operated and to make it more elastic. Network virtualization helps to provision multiple virtual networks (VNs) over a substrate (or physical) network and enables seamlessly sharing of the computing and networking resources in the substrate network [4].

Typically, a VN consists of a set of virtual nodes (*e.g.,* virtual routers) and a few virtual links that connect them. Network virtualization enables a new business model that the infrastructure providers (InPs) provision VN requests from the service providers (SPs), and lease infrastructures to them dynamically for serving the end-users. The best interest of the InPs is to achieve as much revenue as possible from serving the VN requests. Hence, they need to decide how to serve a VN request by allocating appropriate resources in the substrate network, *i.e.*, virtual network embedding (VNE). Specifically, for each VN request, the InP needs to find a set of nodes and links in the substrate network, which have sufficient resources to carry the virtual nodes and virtual links. The VNE problem has been proven to be NP-hard [7], and previous researches have proposed several heuristic algorithms to address it [8–13]. Most of these algorithms performed node mapping based on the local resource information (*i.e.,* the nodes' resources or the nodes' resources together with their incident links' resources). However, this type of approaches could easily result in unbalanced load distribution and congestions in the substrate network. Two recent works in [12, 13] took global resource information into consideration in the node mapping, while the algorithms only adopted over-simplified metrics to measure the embedding capacities of the nodes and did not fully explore the benefit of the global resource information.

In this paper, we propose a novel revenue-driven VNE (*RD-VNE*) algorithm to help InPs maximize their revenue with the assistance of the global resource information. When serving the VN requests, the proposed *RD-VNE* algorithm adopts a node rank, similar to the PageRank in web-search algorithms [14], to rank nodes according to their global resources. We then implement greedy node mapping based on the node ranks, and when node mapping is accomplished, link mapping is performed with the shortest-path routing algorithm. Simulation results show that the proposed *RD-VNE* algorithm outperforms two existing VNE algorithms that also use the global resource information, in terms of request blocking probability and time-average revenue.

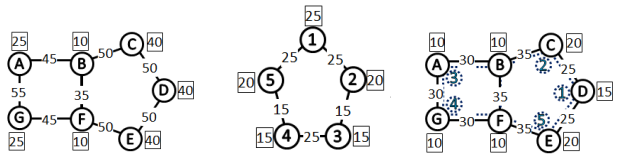
The rest of this paper is organized as follows. We formulate the VNE problem in Section II. The details of the proposed *RD-VNE* algorithm are discussed in Section III. Section IV shows simulation setup and results for the performance evaluation. Finally, Section V summarizes the paper.

II. VNE PROBLEM FORMULATION

A. VNE Models

1) Substrate network: We model the substrate network 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. We denote the available computing resource (*e.g.,* CPU cycles) of node $v^s \in V^s$, and available bandwidth of link $e^s \in E^s$, as c_{v^s} and b_{e^s}, respectively. P_{G^s} is the set of acyclic paths in

978-1-4799-1353-4/13/$31.00 ©2013 IEEE



(a) The substrate network. (b) A VN request. (c) The substrate network after VNE.

Fig. 1. An example of VNE process

$G^s(V^s, E^s)$. An example of a substrate network is illustrated in Fig. 1(a), where the numbers around the nodes and links are their available resources.

2) Virtual network: We use notation ϖ to denote a virtual network (VN) request, whose topology can also be modeled as an undirected graph, $G^r(V^r, E^r)$, where V^r is the set of virtual nodes and E^r is the set of virtual links. Each virtual node $v^r \in V^r$ is associated with a computing resource demand of c_{v^r} and each virtual link $e^r \in E^s$ has a bandwidth demand of b_{e^r}. Fig. 1(b) illustrates a VN request. Each VN request, ϖ, is also associated with two time-domain parameters, *i.e.*, t_ϖ for the arrival time and τ_ϖ for its lifetime.

The notations are summarized in Table I.

TABLE I
NOTATIONS

Notations	Description
G^s	Substrate network
V^s	Set of substrate nodes
E^s	Set of substrate links
c_{v^s}	Available computing resource
b_{e^s}	Available bandwidth
P_{G^s}	Set of acyclic paths
ϖ	VN request
G^r	Virtual network
V^r	Set of virtual nodes
E^r	Set of virtual links
c_{v^r}	Computing resource demand
b_{e^r}	Bandwidth demand
t_ϖ	Arrival time of ϖ
τ_ϖ	Lifetime of ϖ

B. VNE Process

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

1) Node Mapping: The InP finds, for each node from the VN request, a unique substrate node that has enough available computing resource to meet its computing resource demand, through a 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 constraints:

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

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

[1]Note that virtual nodes from different VN requests can be mapped onto the same substrate node as long as the available computing resource is sufficient.

2) Link Mapping: For two adjacent nodes in the VN request, the InP finds one or more paths between the two mapped substrate nodes, and the total bandwidth of the path(s) should be larger than the corresponding virtual link demand. Specifically, the mapping is $F_L : E^r \mapsto P_{G^s}$, such that,

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

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

$$b_{e^r} \leq B_{F_L(e^r)}, \tag{3}$$

where, $B_{F_L(e^r)}$ is the total available bandwidth of path set $F_L(e^r)$[2].

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

C. VNE Revenue Model

In this paper, we adopt a "pay-per-user" revenue model, based on the "on-demand" cloud service price scheme by Amazon Web Services (AWS) [16]. Specifically, for a VN request, 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 from ϖ. $R_0(\varpi)$ 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. In practice, the InP aims to maximize its revenue, under the resource limitation of its substrate network, by adopting different VNE policies.

III. RD-VNE ALGORITHM

In this section, we propose a novel VNE algorithm. The algorithm implements a PageRank-like node-ranking approach based on the global resource information, and then performs node mapping according to the ranks of the nodes. After node mapping, the shortest-path based link mapping follows. We refer this algorithm as revenue-driven VNE (*RD-VNE*) algorithm, for it aims to maximize the InP's revenue by accepting as many as possible VN requests under the resource limitation of its substrate network.

A. Node Ranking

For the node ranking, we adopt a novel ranking approach, similar to the PageRank algorithm [14], to estimate the embedding capability of each substrate node for carrying the VN request. The rank of each node considers the global resource information in a recursive manner. Specifically, for a network

[2]Note that, $B_{F_L(e^r)}$ can be calculated by *Ford-Fulkerson* algorithm [15].

topology $G(V, E)$ (G can be either a substrate network or a VN topology), the node rank is calculated as follows,

$$\Upsilon_u = (1-d) \cdot c'_u + d \sum_{v \in N(u)} \frac{b_{(u,v)} \cdot \Upsilon_v}{\sum_{w \in N(v)} b_{(w,v)}} \quad (6)$$

where Υ_u is the node rank of node $u \in V$, d is a damping factor within $(0, 1)$, $N(v)$ indicates the set of nodes that connect to node $v \in V$ directly, namely, $N(v) = \{u : (u, v) \in E\}$, and $b_{(u,v)}$ represents the available bandwidth (or bandwidth demand) on the link $(u, v) \in E$. Moreover, c'_u is the normalized available computing resource (or computing resource demand) on node $u \in V$, defined as:

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

Specifically, the node rank increases with the node's available computing resource and the embedding capacities of its direct-neighbors, since higher ranked substrate nodes can achieve larger successful probability in the link-mapping step.

Writing in the matrix form, we can represent the node ranks of all the nodes in a vector as follows,

$$\boldsymbol{\Upsilon} = (1-d)\mathbf{c} + d\mathbf{M}\boldsymbol{\Upsilon}, \quad (7)$$

where $\boldsymbol{\Upsilon} = (\Upsilon_1, \Upsilon_2, ..., \Upsilon_{|V|})^T$, $\mathbf{c} = (c'_1, c'_2, ..., c'_{|V|})^T$, and $\mathbf{M}$ is a transition matrix of dimension $|V| \times |V|$. Each element in $\mathbf{M}$, following from Eqn. (6), is defined as,

$$m_{(u,v)} = \begin{cases} \frac{b_{(u,v)}}{\sum_{x \in N(v)} b_{(x,v)}} & (u, v) \in E \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

We first prove the uniqueness of the node-rank vector.

Proposition 3.1: Matrix $\mathbf{I} - d\mathbf{M}$ is reversible. It concludes that a unique vector $\boldsymbol{\Upsilon}$ can be obtained from Eqn. (7).

Proof: With Eqn. (8), we have

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

According to the Gershgorin Circle Theorem [17], we can conclude that $\|\mathbf{M}\| \leq 1$.

Suppose that the matrix $\mathbf{I} - d\mathbf{M}$ is singular, then the linear system of equations $(\mathbf{I} - d\mathbf{M})\mathbf{x} = 0$ has non-zero solutions. Let $\mathbf{x}_0$ be a non-zero solution of this linear system of equations, then $d\mathbf{M} \cdot \mathbf{x}_0 = \mathbf{x}_0$. Thus, $\|\mathbf{x}_0\| = \|d\mathbf{M} \cdot \mathbf{x}_0\| \leq d\|\mathbf{M}\| \cdot \|\mathbf{x}_0\|$. As a result $\|\mathbf{M}\| \geq \frac{1}{d} > 1$. Obviously, it is contradicted with $||\mathbf{M}|| \leq 1$. Therefore, we can conclude that matrix $\mathbf{I} - d\mathbf{M}$ is reversible. ■

Obviously, the unique solution of Eqn. (7) is given by

$$\boldsymbol{\Upsilon} = (1-d)(\mathbf{I} - d\mathbf{M})^{-1}\mathbf{c}. \quad (9)$$

The complexity of calculating the node-rank vector with Eqn. (9) directly is $O(|V|^3)$. When the size of the network topology gets large, the calculation could be time consuming. We develop a simple iterative calculation strategy as follows,

$$\boldsymbol{\Upsilon}_{k+1} = (1-d)\mathbf{c} + d\mathbf{M}\boldsymbol{\Upsilon}_k, \quad (10)$$

where $\boldsymbol{\Upsilon}_k$ is the node-rank vector after k iterations. *Algorithm 1* shows the details of the node ranking. The complexity of the algorithm is $O(|V|^2 log(\frac{1}{\delta}))$ [18]. It is easy to prove that this algorithm could always converge to the solution of Eqn. (7), as the procedures are equivalent to the Jacobi algorithm [17] for solving the linear system of equations .

Algorithm 1: Node Ranking based on Global Resource Information

input : Network topology $G(V, E)$, a small enough positive real number δ
output: Node-rank vector $\boldsymbol{\Upsilon}$

```
calculate matrix M and vector c respectively;
Υ_0 = c;
k = 0;
Δ = ∞;
while Δ ≥ δ do
    Υ_{k+1} = (1 − d)c + dM · Υ_k;
    Δ = ‖Υ_{k+1} − Υ_k‖;
    k = k + 1;
end
Υ = Υ_k;
```

B. Node Mapping

In the proposed *RD-VNE* algorithm, the node mapping works as follows in a greedy way. It first backups the status of the substrate network, and then sorts the nodes of both the substrate network and the VN request in the descending order according to the node-rank vectors calculating by *Algorithm 1*. As the value of the node rank indicates the embedding capacity of the corresponding node, from the perspective of load balancing, we embeds the virtual nodes onto the substrate nodes by processing the two sorted node lists with a strategy similar to the *merge-sort* algorithm [19]. Thus, the virtual node with the highest node rank among the remaining ones will always embed onto the substrate node that also has the highest node rank among the remaining substrate ones, whose available computing resource meets the demand. If the computing resource demand cannot be satisfied by any of the remaining substrate nodes, the VN request is marked as blocked, and if all virtual nodes are embedded successfully, the computing resources of the corresponding substrate nodes would be updated. The details of the greedy node mapping algorithm are shown in *Algorithm 2*. The complexity of this algorithm is $O(|V^s||V^r|)$.

C. Shortest-Path based Link Mapping

For link mapping, we apply the shortest-path routing algorithm, which aims to minimize the total substrate bandwidth allocated to each virtual link. Specifically, we embed the virtual links of the VN request one by one, and for each virtual link, we adopt the Dijkstra's algorithm to find the shortest path between the two corresponding nodes in the substrate network. What's more, to improve the efficiency of

Algorithm 2: Greedy Node Mapping

input : VN request ϖ, $G^r(V^r, E^r)$, substrate network $G^s(V^s, E^s)$, the node-rank vector $\boldsymbol{\Upsilon}^s$ for substrate network, and the node-rank vector $\boldsymbol{\Upsilon}^r$ for ϖ
output: Node mapping F_N, and backup substrate network status G_0^s

backup G^s into G_0^s;
$F_N \leftarrow \mathbf{0}$;
$Count = 0$;
get $\boldsymbol{\Upsilon}^s_{sort}$ by sorting $\boldsymbol{\Upsilon}^s$ in descending order;
get $\boldsymbol{\Upsilon}^r_{sort}$ by sorting $\boldsymbol{\Upsilon}^r$ in descending order;
for *each virtual node* v^r *in the order of* $\boldsymbol{\Upsilon}^r_{sort}$ **do**
  for *each remaining substrate node* v^s *in the order of* $\boldsymbol{\Upsilon}^s_{sort}$ **do**
    if $c_{v^s} \geq c_{v^r}$ **then**
      $F_N(v^r) = v^s$;
      $Count = Count + 1$;
      break;
    end
  end
  if $F_N(v^r) = 0$ **then**
    break;
  end
end
if $Count = |V^r|$ **then**
  update computing resources of V^s in G^s;
  continue with link mapping;
else
  mark ϖ as blocked;
  wait for the next VN request;
end

Algorithm 3: Shortest-Path based Link Mapping

input : VN request ϖ, $G^r(V^r, E^r)$, substrate network $G^s(V^s, E^s)$, node mapping F_N, and backup substrate network status G_0^s
output: Link mapping F_L

$F_L \leftarrow \mathbf{0}$;
$flag \leftarrow$ FALSE;
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;
    flag $\leftarrow$ TRUE;
    break;
  else
    update bandwidth of E^s in G^s;
    $F_L(e^r) \leftarrow p^s_{e^r}$;
  end
end
if $flag$ **then**
  restore G^s to G_0^s;
end
wait for the next VN request;

the algorithm, we propose a pruning optimizer, which precuts all the substrate links in the substrate network that do not have enough bandwidth for the corresponding virtual link. If the link mapping fails, *i.e.,* not all virtual links are embedded successfully, we restore the status of the substrate network and mark the VN request as blocked. *Algorithm 3* describes the details of the shortest-path based link mapping algorithm. The complexity of this algorithm is $O(|E^r||E^s|log|V^s|)$.

D. Time Complexity

The time complexity of the proposed *RD-VNE* algorithm can be calculated by adding up the complexities of its three steps, *i.e.*, node ranking, node mapping and link mapping. Hence, we determine that its time complexity is $O((|V^s|^2 + |V^r|^2) \cdot log(\frac{1}{\delta}) + |V^s||V^r| + |E^r||E^s|log|V^s|)$.

IV. Performance Evaluation

A. Simulation Setup

To evaluate the proposed *RD-VNE* algorithm, we perform extensive simulations with a substrate topology that is randomly generated by the GT-ITM tool [20]. The initial available computing resource and bandwidth of the substrate network are randomly selected by uniform distribution within the ranges shown in Table II. The topologies of the VN requests are also randomly-generated by the GT-ITM tool. For each VN request, the computing resource demand and the bandwidth demand are uniformly distributed within the ranges shown in Table II. The number of virtual nodes in each VN request is selected from 2 to 20 following the discrete uniform distribution, and each node-pair is connected with certain probability (*i.e.*, link connectivity rate). The arrival of the VN requests follows the Poisson traffic model. Assume that the average arrival rate is λ VN requests per time-unit, and the average lifetime of the VN requests is $\frac{1}{\mu}$ time-units, then we could use $\lambda \cdot \frac{1}{\mu}$ (Erlangs) to quantify the traffic load of the VN requests. Table II shows the simulation parameters.

B. Performance metrics

Similar to previous works [11, 12], we consider two performance metrics, including the blocking probability and the time-average revenue, whose definitions are as follows,

- Blocking probability:

$$p_b = \lim_{T \to \infty} \frac{\mathcal{N}_b(T)}{\mathcal{N}(T)}, \tag{11}$$

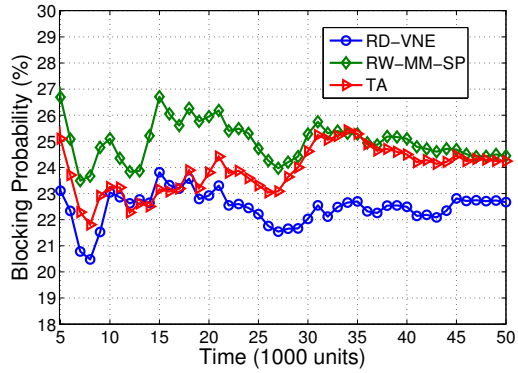


Fig. 2. Blocking probabilities in a long run (25 Erlangs).

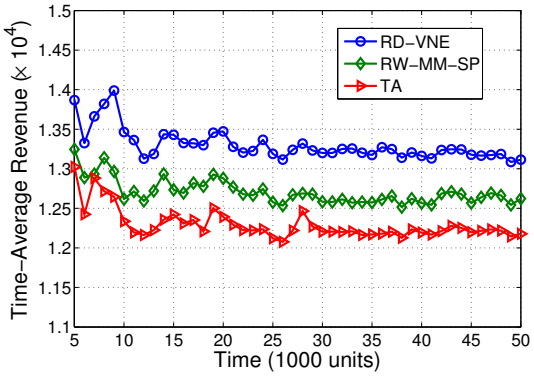


Fig. 3. Time-average revenues in a long run (25 Erlangs).

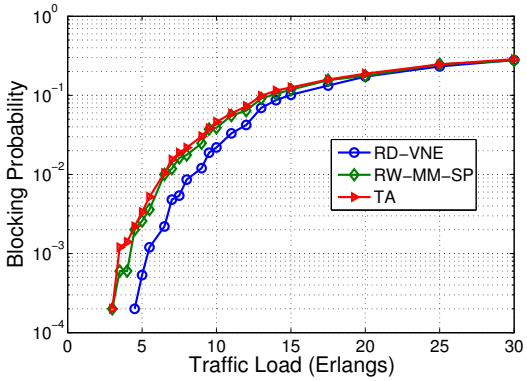


Fig. 4. Blocking probabilities for different traffic loads.

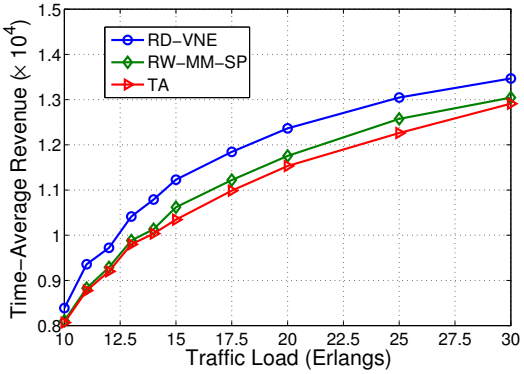


Fig. 5. Time-average revenues for different traffic loads.

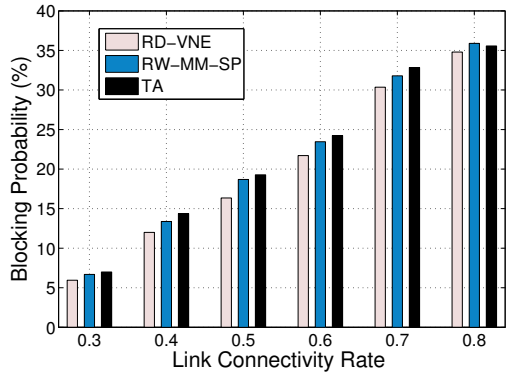


Fig. 6. Blocking probabilities for different link connectivity rates.

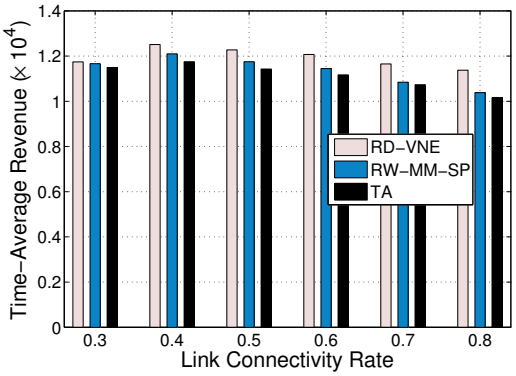


Fig. 7. Time-average revenues for different link connectivity rates.

where, $\mathcal{N}_b(T)$ and $\mathcal{N}(T)$ are the numbers of blocked (or rejected) VN requests and total VN requests in time period T, respectively.

- Time-average revenue:

$$R_{G^s} = \lim_{T\to\infty} \frac{\sum_{\varpi\in\Phi_T} R(\varpi)}{T}, \tag{12}$$

where $\Phi_T = \{\varpi : 0 \leq t_\varpi \leq T\}$ denotes the set of VN requests arriving before time instance T.

C. Performance Comparisons

In this subsection, we compare the performance of our proposed *RD-VNE* algorithm with two existing global resource information based VNE algorithms, *i.e.,* the *RW-MM-SP* algorithm [12] and the *TA* algorithm [13]. For all simulations, we set: $\alpha = \beta = 1$, $\delta = 0.00001$, and $d = 0.85$. All the parameters of *RW-MM-SP* and *TA* are from [12] and [13], respectively. Figs. 2-5 are from the simulations using VN requests whose link connectivity rate is 0.5, while Figs. 6-7 are from those that have VN requests with different link connectivity rates.

1) Blocking probability comparisons: Figs. 2, 4 and 6 show that *RD-VNE* provides the lowest blocking probabilities. We believe that the lower blocking probability of the *RD-VNE* results from the more efficient node-ranking approach, which

TABLE II
SIMULATION PARAMETERS

Substrate network	Number of nodes	100
	Number of links	570
	Minimum node degree	4
	Maximum node degree	20
	Initial available computing resource	50-100 units
	Initial available bandwidth	50-100 units
VN requests	Number of nodes	2-20
	Link connectivity rate	0.3-0.8
	Average lifetime	500 time-units
	Computing resource demand	0-50 units
	Bandwidth demand	0-50 units

could fully explore the benefit of the global resource information. Thus, the *RD-VNE* algorithm makes better utilization of the substrate resources and leaves more resources for future VN requests, which in turn brings down the request blocking probability.

2) Time-average revenue comparisons: Figs. 3, 5 and 7 show the proposed *RD-VNE* algorithm provides the highest time-average revenue among the three VNE algorithms. These results verify that the *RD-VNE* not only accepts more VN requests, but also brings more revenue to the InP. Therefore, the lower blocking probabilities of the *RD-VNE* in Figs. 2, 4 and 6 were not achieved by accepting the "small-revenue" VN requests, while rejecting the "big-revenue" ones.

From Fig. 5, we can also observe that the time-average revenue of all the three algorithms increase with the traffic load in a concave manner. This observation can be understood as follows. When the traffic load increases, more VN requests will exist in the substrate network to share the resources of the substrate network. As the resources in the substrate network is not infinite, the substrate network could get saturated. And when the substrate network is getting saturated, it tends to use more resources on average to accept the VN requests, *e.g.*, the shortest path between two substrate nodes is congested and we have to switch to a longer path for the link mapping. Consequently, the increase of the time-average revenue would become slower as the traffic load increases further. This rationale could be verified by the results in Fig. 4, where the blocking probabilities of the three algorithms finally get merged, when the traffic load gets higher.

In Fig. 7, we find that the time-average revenue first increases and then decreases as the link connectivity rate increases. This interesting phenomenon can be explained as follows. As the link connectivity rate increases, the average revenue per VN request also increases, while it also becomes harder to embed each VN request successfully, and this makes the blocking probability increase monotonically. Therefore, at certain point, the penalty due to the blocking probability increase would conquer the revenue gain brought in by each successfully served VN request.

V. CONCLUSION

In this paper, we proposed a novel *RD-VNE* algorithm based on the global resource information. The proposed algorithm introduced a node-ranking approach, based on the *PageRank* Algorithm, to rank all nodes based on their capacities for serving VN requests, with the help of the global resource information. The new ranking approach was then used to assist the greedy node mapping, and the link mapping is based on the shortest-path routing. The simulation results showed that the proposed *RD-VNE* algorithm outperformed two existing VNE algorithms that were also based on the global resource information, in terms of request blocking probability and time-average revenue.

ACKNOWLEDGMENT

This work was supported in part by the New Century Excellent Talents (NCET) in University Program under Project NCET-11-0884.

REFERENCES

[1] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *IEEE Comput.*, vol. 38, pp. 34–41, Apr. 2005.
[2] J. Turner and D. Taylor, "Diversifying the Internet," in *Proc. of 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] A. Nakao, "Network virtualization as foundation for enabling new network architectures and applications," *IEICE Trans. Commun.*, vol. E93-B, pp. 454–457, Mar. 2010.
[5] N. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Netw.*, vol. 54, no. 5, pp. 862–876, Apr. 2010.
[6] A. Khan, A. Zugenmaier, D. Jurca, and W. Kellerer, "Network virtualization: a hypervisor for the internet?" *IEEE Commun. Mag.*, vol. 50, no. 1, pp. 136–143, 2012.
[7] D. G. Andersen, "Theoretical approaches to node assignment," Dec. 2002, unpublished Manuscript.
[8] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proc. of INFOCOM 2006*, pp. 1 –12, Apr. 2006.
[9] J. Lu and J. Turner, "Efficient mapping of virtual networks onto a shared substrate," Washington University in St. Louis, Tech. Rep., 2006.
[10] 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.
[11] N. Chowdhury, M. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. of INFOCOM 2009*, pp. 783 –791, Apr. 2009.
[12] X. Cheng *et al.*, "Virtual network embedding through topology-aware node ranking," *SIGCOMM Comput. Commun. Rev.*, vol. 41, pp. 39–47, Apr. 2011.
[13] S. Zhang, Z. Qian, J. Wu, and S. Lu, "An opportunistic resource sharing and topology-aware mapping framework for virtual networks," in *Proc. of INFOCOM 2012*, pp. 2408 –2416, Mar. 2012.
[14] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," in *Proc. of WWW 1998*, pp. 107–117, 1998.
[15] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows: theory, algorithms, and applications*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.
[16] [Online]. Available: http://aws.amazon.com/ec2/#pricing
[17] G. H. Golub and C. F. van Van Loan, *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)(3rd Edition)*, 3rd ed. Johns Hopkins University Press, Oct. 1996.
[18] M. Bianchini, M. Gori, and F. Scarselli, "Inside PageRank," *ACM Trans. Internet Technol.*, vol. 5, no. 1, pp. 92–128, 2005.
[19] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*, 2nd ed. McGraw-Hill Higher Education, 2001.
[20] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. of INFOCOM 1996*, pp. 594–602, Mar. 1996.