Virtual Network Embedding Through Topology-Aware Node Ranking

Xiang Cheng, Sen Su[*], Zhongbao Zhang, Hanchi Wang, Fangchun Yang
Beijing University of Posts and Telecommunications
{chengxiang,susen, zhongbaozb, luigiking, fcyang}@bupt.edu.cn

Yan Luo, Jie Wang
University of Massachusetts Lowell
yan_luo@uml.edu, wang@cs.uml.edu

ABSTRACT

Virtualizing and sharing networked resources have become a growing trend that reshapes the computing and networking architectures. Embedding multiple virtual networks (VNs) on a shared substrate is a challenging problem on cloud computing platforms and large-scale sliceable network testbeds. In this paper we apply the Markov *Random Walk* (RW) model to rank a network node based on its resource and topological attributes. This novel topology-aware node ranking measure reflects the relative importance of the node. Using node ranking we devise two VN embedding algorithms. The first algorithm maps virtual nodes to substrate nodes according to their ranks, then embeds the virtual links between the mapped nodes by finding shortest paths with unsplittable paths and solving the multi-commodity flow problem with splittable paths. The second algorithm is a backtracking VN embedding algorithm based on breadth-first search, which embeds the virtual nodes and links during the same stage using node ranks. Extensive simulation experiments show that the topology-aware node rank is a better resource measure and the proposed RW-based algorithms increase the long-term average revenue and acceptance ratio compared to the existing embedding algorithms.

Categories and Subject Descriptors

C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks; G.1.6 [**Numerical Analysis**]: Optimization

General Terms

Algorithms; Design; Performance

Keywords

Network Virtualization; Cloud Computing; Virtual Network Embedding; Topology-aware; Random Walk; Markov Chain

1. INTRODUCTION

Sharing virtualized resources enables new computing and networking paradigms such as cloud based computing platforms [1] and sliceable network testbeds [15]. Users of a cloud platform or a network infrastructure request their share of resources including CPU capacities, storage space, network bandwidth, etc., while the infrastructure providers make their best effort to serve the requests, which are also known as virtual networks (VNs). The allocation of resources to VNs in such a virtualization environment is critical to both users' computation needs and the resource providers' monetary gain.

In the multi-tenant network virtualization environments, infrastructure providers (InPs) (e.g., cloud providers) and service providers (SPs) (e.g., cloud users/tenants) play two decoupled roles, namely, InPs manage the physical infrastructure while SPs create VNs and offer end-to-end services [27, 14, 8]. Mapping VN requests of the SPs onto the substrate network of the InPs, also known as VN embedding, is NP-hard[28, 5]. Thus, devising heuristics has become the main line of research in VN embedding [13, 29, 22, 28]. The early algorithms measure the resource of a node by its CPU capacity, or bandwidth, or both, without considering the topological structure of the VNs and the underlying substrate network. Yet the topological attributes of nodes have significant impact on the success and efficiency of mapping outcomes. It would make sense to measure a node's resources and its topological attributes at the same time.

Inspired by PageRank used by Google's search engine, which measures the popularity of web pages based on Markov random walks, we use the same theory to measure topology-aware resource ranking of a node, called NodeRank, which reflects the resource and quality of connections of a node. PageRank considers a link from page A to page B as a vote, and a page is considered important if a number of important pages vote to it. In such a way, the topology of the world wide web influences the PageRank of a web page. In VN embedding, if a node links forward to a number of nodes with relatively high importance, this node would also be considered important, where the importance refers to the relative resource quality of a node. We will take into account not only the availability or requirements of the CPU and link resources of the node, but also its topological characteristics, i.e., the quality of its neighbors. Treating the connectivity between two nodes as a Markov chain transition with certain probability, we can calculate the relative resource quality of a node with a Markov chain model based on the topology of the network.

We devise two new VN embedding algorithms called *RW-MaxMatch* and *RW-BFS* based on NodeRanks. They first compute the node rank for each node in the VN request and for each node in the residual substrate network. *RW-MaxMatch* is a two-stage VN embedding algorithm. In the first stage it maps a virtual node with the highest rank to

[*]Corresponding author of this paper is Prof. Sen Su

a substrate node with the highest rank, a virtual node with the second highest rank to a substrate node with the second highest rank, and continues in this manner for the rest of the virtual nodes. In the second stage it embeds virtual links using the shortest path algorithm if path splitting [28] is not supported by the substrate network, or using the multi-commodity flow algorithm if the substrate supports path splitting. Similar to the existing two-stage VNE algorithms, RW-MaxMatch may lead to higher substrate network resource consumption and restrict the ability of the substrate to accept additional future requests. *RW-BFS* can help solve this problem. It is a backtracking VN embedding algorithm based on breadth-first search that maps virtual nodes and virtual links during the same stage, aiming to increase the resource utilization of the substrate resource. Extensive simulation experiments show that the topology-aware node rank is a better node resource measure and the proposed RW-based algorithms increase the long-term average revenue and acceptance ratio compared to existing embedding algorithms.

This paper presents the following major contributions:

- We formulate a Markov random walk model to compute topology-aware resource ranking of nodes in a network, which serves as the basis of embedding virtual networks on substrate networks. To the best of our knowledge, this work is the first to apply random walks in solving VNE problems.
- We devise two VNE algorithms based on topology-aware node ranks. Both two-stage and one-stage mapping strategies are investigated.
- We conduct a thorough comparison between our algorithms and a wide range of existing algorithms through extensive simulations. We design a VNE simulator and make it publicly available to the research community.

The rest of the paper is organized as follows. In Section 2, we discuss the related work. Section 3 presents the network model and formalizes the VN embedding problem. In Section 4 we present the method of computing the topology-aware resource ranks of nodes using the random walk model. Section 5 describes RW-MaxMatch and RW-BFS. The VN embedding algorithms are evaluated in Section 6. Section 7 concludes the paper.

2. RELATED WORK

The VN embedding problem is similar to the virtual private network (VPN) provisioning problem [18]. The major difference between them is on resource constraints. In a typical VPN request, the only resource constraints are bandwidth requirements from sources to destinations specified by a traffic matrix. There are typically no resource constraints on the nodes (e.g., CPU) and their locations. Another similar problem is the network testbed mapping problem. The *Assign* algorithm [24] used in the Emulab testbed considers constraints on both nodes and links, where the node constraint is provided as the exclusive use of nodes, i.e., different virtual networks cannot share the same substrate node. VN embedding, however, allows substrate nodes and links to be shared by multiple VNs.

Early studies on VN embedding either assume that the VN requests are known in advance (an offline version) [29, 22]; or deal with at most one type of constraints (node or link) [13, 29, 22]; or perform no admission control when the resource of the substrate network is insufficient [13, 29, 22]; or focus only on the backbone-star topology [22].

Without reducing the problem space, Yu et al. [28] introduce the mechanisms of substrate supporting path splitting and migration. Chowdhury et al. [9], while considering the same online VN embedding problem space as in [28], also consider location requirements of virtual nodes and use mixed integer programming (MIP) to solve the VN embedding problem.

Lischka et al. [21] model the topology of the substrate and the virtual network as a directed graph, and propose a VN embedding algorithm based on subgraph isomorphism which maps nodes and links during the same stage. Their algorithm can be seen as a extended version of the classic VF graph matching algorithms [10], where link-on-link mapping has been relaxed.

Houidi et al. [20] present a distributed VN embedding algorithm that achieves embedding through communicating and exchanging messages between agent-based substrate nodes. Although centralized algorithms could suffer from a single point of failure, the performance and scalability of the proposed distributed algorithm compare unfavorably with those of the centralized algorithms.

To maximize the aggregate performance across virtual networks, He et al. [19] propose an architectural framework called DaVinci to dynamically adapt virtual networks for a customized network substrate, where each substrate link periodically reassigns bandwidth among its virtual links. While on a smaller timescale, a distributed protocol is run in each VN to maximize the VN's own performance objective independently. DaVinci, however, does not have a node embedding stage.

Since the network condition change over time due to the arrival and departure of VNs, resources in the substrate network may become fragmented. Butt et al. [7] present a topology-aware measure using scaling factors for the substrate network, which identifies the bottleneck nodes and links in the substrate network. They then propose a set of algorithms for re-optimizing and re-embedding initially-rejected VN requests.

Recently, Guo et al. proposed a data center network virtualization architecture called SecondNet[17]. In SecondNet, the unit of resource allocation for multiple tenants in the cloud is referred as virtual data center (VDC) which consists of virtual machines and virtual links. The VDC resource allocation problem is close related to the VN embedding problem and the main difference is the problem scale. The VDC resource allocation algorithms proposed in [17] primarily focus on how to quickly allocate the resources to the VDCs when a VDC has thousands of virtual machines and the cloud infrastructure has tens to hundreds of thousands servers and switches, and how to satisfy the elasticity requirement of VDCs. We believe that the topology-aware node ranking method is general enough to be applied in their context to increase the possibility of satisfying the resource requirements of VDCs.

Page et al. [23] use random walks to rank the relative importance of web pages, where the rank of a page depends on the topological properties of the weighted links between the pages, regardless of their content. A more general framework for this scheme was proposed in [11]. Another example

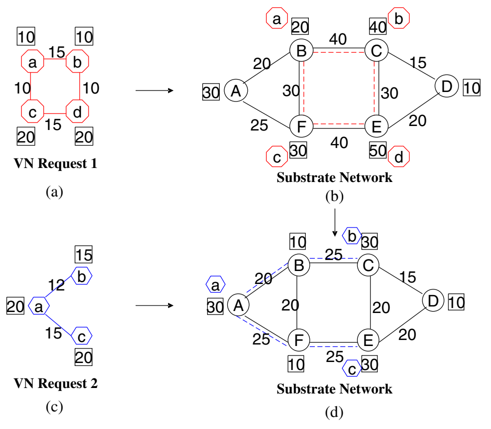


Figure 1: Examples of VN embedding

of using random walks is to compute a set of topological signatures for each node in a graph, and it is also shown to be effective for exact (and approximate) graph matching (see [16]). In a typical graph matching problem, there are either no weights on the nodes or links, or there are only visual features (e.g., RGB color space) contained in the nodes without links. Unlike PageRank and graph matching, however, in the VN embedding problem there are weights on both nodes and links, and the weights are typically non-uniform.

Our work differs from the existing studies in three ways. First, we address the online VN embedding problem with admission control, and do not need to reduce problem space as in [29, 22, 13]. Second, we consider both the resource amount and topology properties of a node in a unified way to rank the relative importance of a node, which will be leveraged in the mapping procedure. Different from existing work that only takes into consideration the resource (e.g., CPU, bandwidth, or both) of a node while neglecting its topology property in computing the resource availability, our work mends this gap. Third, our topology-aware node ranking measure focuses on leveraging such a rank to benefit the current VN embedding process rather than identifying the bottlenecks of the substrate nodes and links for the VN embedding reoptimization process proposed in [7]. In Section 4 we will provide details how we apply the random walk model to compute the topology-aware node resource ranks.

3. NETWORK MODEL AND PROBLEM DESCRIPTION

Substrate Networks. A substrate network can be represented by a weighted undirected graph $G_s = (N_s, L_s, A_s^n, A_s^l)$, where N_s is the set of substrate nodes and L_s the set of substrate links. The notations A_s^n and A_s^l denote the attributes of the substrate nodes and links, respectively. The attributes of the node include processing capacity, storage, and location. The typical attribute of the link is its bandwidth. In this paper we consider the available CPU capacity for the node attribute and the available bandwidth for the link attribute as in most of the previous research. Denote by P_s the set of all loop-free paths of the substrate network. Fig. 1(b) presents a substrate network, where the numbers in rectangles are the available CPU resources at the nodes and the numbers over the links represent available bandwidths.

Virtual Network Request. Similar to the substrate network, we use an undirected graph $G_v = (N_v, L_v, C_v^n, C_v^l)$ to denote a virtual network, where N_v is the set of virtual nodes and L_v the set of virtual links. Virtual nodes and links are associated with their capacity constraints, denoted by C_v^n and C_v^l, respectively. We also denote a VN request by $VNR^{(i)}(G_v, t_a, t_d)$, where t_a is the arrival time of the VNR and t_d the duration of the VN staying in the substrate network. When the i-th VNR arrives, the substrate network should allocate resources to the VN to meet the requirements of the virtual nodes and links. If there are no sufficient substrate resources available, the VNR should be rejected or postponed. The allocated substrate resources are released when the VN departs. There may be different roles of all node in using VNs, such as directory or file server etc. For simplicity, like most of the previous work [28, 9], we ignore these dependencies. Fig. 1(a) and Fig. 1(c) present two VN requests with node and link requirements.

VN Embedding Problem Description. The VN embedding problem is defined by a mapping $M : G_v(N_v, L_v) \rightarrow G_s(N_s^{'}, P_s^{'})$ from G_v to a subset of G_s, where $N_s^{'} \subset N_s$ and $P_s^{'} \subset P_s$. The mapping can be decomposed into two mapping steps: (i) node mapping places the virtual nodes to different substrate nodes that satisfy the node resource constraints; and (ii) link mapping assigns the virtual links to loop-free paths on the substrate that satisfy the link resource requirements.

Fig. 1(a) and Fig. 1(b) show a VN embedding solution for VNR 1. Fig. 1(c) and Fig. 1(d) show another VN embedding solution for VNR 2, where residual resources are also shown. Note that the virtual nodes of different VNRs can be mapped onto the same substrate node.

Objectives. The main objective of VN embedding is to map the VNs to the substrate network to make efficient use of the substrate network resources, when the VN requests arrive and depart over time.

Similar to the previous work in [29, 28, 9], the revenue of accepting a VNR at time t can be formulated by

$$R(G_v, t) = \sum_{d_v \in N_v} CPU(d_v) + \sum_{l_v \in L_v} BW(L_v), \qquad (1)$$

where $CPU(d_v)$ and $BW(L_v)$ are the CPU and the bandwidth requirements for virtual node d_v and link l_v, respectively.

The cost of accepting a VNR at time t is defined as the sum of the total substrate resources allocated to that VN:

$$C(G_v, t) = \sum_{d_v \in N_v} CPU(d_v) + \sum_{l_v \in L_v} \sum_{l_s \in L_s} BW(f_{l_s}^{l_v}, l_v), \quad (2)$$

where $f_{l_s}^{l_v} \in \{0, 1\}$ and $f_{l_s}^{l_v} = 1$ if substrate link l_s allocated bandwidth resource to virtual link l_v, otherwise $f_{l_s}^{l_v} = 0$. $BW(f_{l_s}^{l_v}, l_v)$ is the bandwidth allocated to l_v from l_s.

From the InPs' point of view, an efficient and effective online VN embedding algorithm would maximize the revenue of InPs and increase the utilization of the substrate network in the long run. Like the previous work in [28], the long-term average revenue is given by

$$\lim_{T \to \infty} \frac{\sum_{t=0}^{T} R(G_v, t)}{T}. \qquad (3)$$

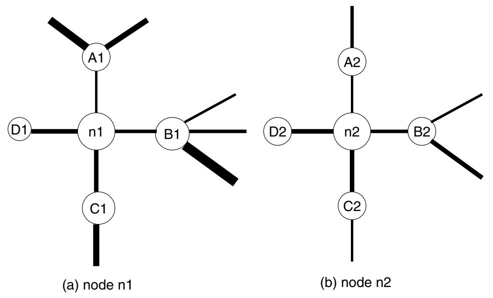


Figure 2: Motivational example

The VNR acceptance ratio of the substrate network can be defined by

$$\lim_{T\to\infty} \frac{\sum_{t=0}^{T} VNR_s}{\sum_{t=0}^{T} VNR}, \tag{4}$$

where VNR_s is the number of VN requests successfully accepted by the substrate network.

We also consider the long-term revenue to cost ratio to quantify the efficiency of resource utilization of the substrate network:

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

If the long-term average revenues of the VN embedding solutions are about the same, the higher VN acceptance ratio and R/C ratio are preferred.

4. TOPOLOGY-AWARE NODE RANKING

VN embedding incurs node mapping and link mapping. Node mapping can be achieved by selecting substrate nodes with sufficient CPU resources, and link mapping requires sufficient link resource on both of the selected nodes and the path between any two selected substrate nodes. Most early publications (e.g., [28]) perform node mappings and link mappings at two different stages, where nodes are selected first at the node-mapping stage, and link allocation and path selection are done at the link-mapping stage. We take a different approach by incorporating topology attributes during the node mapping stage, aiming to improve the success rate and efficiency of link mapping. A motivational example is illustrated in Fig. 2, where larger nodes are nodes with more CPU resources and the wider lines are links with more bandwidth resources. Nodes $n1$ and $n2$ seem to have the same resource availability if they are considered alone. However, $n1$ is a "better" node because the neighbors of $n1$, namely, $A1$, $B1$, and $C1$, have more resources than those of $n2$'s neighbors, and so mapping a virtual node to $n1$ has a higher chance to achieve a successful link mapping.

We define the notion of node rank to measure the resource availability of a node. Intuitively, the rank of a given node u is determined by its CPU power and its collective bandwidth of outgoing links. It is also affected by the ranks of the nodes that can be reached from u. We model the first using the product of its CPU and collective bandwidth of outgoing links as in [28]. We model the second by dividing reachable nodes into two groups, that is, the nodes that are incident to the outgoing links from u and the nodes that can be reached from u via multiple hops. Modeling connectivity is challenging, and in this paper we define a jumping probability to model the likelihood of a node that is reachable from u via multiple hops and define a forward probability to model the influence of the neighboring nodes from u's forward links. In particular, let

$$H(u) = CPU(u) \sum_{l\in L(u)} BW(l), \tag{6}$$

where, on a substrate network, $L(u)$ is the set of all the outgoing links of u, $CPU(u)$ is the remaining CPU resource of u, and $BW(l)$ is the unoccupied bandwidth resource of link l. On a virtual node, $CPU(u)$ and $BW(l)$ are the capacity constraints of the node u, respectively. The initial NodeRank value for node u can be computed by

$$NR^{(0)}(u) = \frac{H(u)}{\sum\limits_{v\in V} H(v)}. \tag{7}$$

Let $u, v \in V$ be two different nodes. Let

$$p_{uv}^{J} = \frac{H(v)}{\sum\limits_{w\in V} H(w)}, \tag{8}$$

$$p_{uv}^{F} = \frac{H(v)}{\sum\limits_{w\in nbr_1(u)} H(w)}, \tag{9}$$

where p_{uv}^{J} denotes the jumping probability from node u to land on node v, $nbr_1(u) = \{v \mid (u,v) \in E\}$, and p_{uv}^{F} the forward probability from node u to node v with $(u,v) \in E$. Clearly,

$$\sum_{v\in V} p_{uv}^{J} = 1, \quad \sum_{v\in nbr_1(u)} p_{uv}^{F} = 1.$$

The probabilities p_{uv}^{J} and p_{uv}^{F} may be viewed as resource voting for node u from, respectively, any node reachable from u and node u's neighboring nodes. The voting from a non-neighboring node implies that there should exist a multi-hop path between u and v. Thus, the topology information of the two nodes is also embedded in the probabilities.

For any node $v \in V$, let

$$NR^{(t+1)}(v) = \sum_{u\in V} p_{uv}^{J} \cdot p_{u}^{J} \cdot NR^{(t)}(u) + \sum_{u\in nbr_1(v)} p_{uv}^{F} \cdot p_{u}^{F} \cdot NR^{(t)}(u), \tag{10}$$

where $p_u^J + p_u^F = 1$, $p_u^J \geq 0$, $p_u^F \geq 0$, and $t = 0, 1, \cdots$. The p_u^J and p_u^F are bias factors, and we will typically want to set p_u^J to 0.15 and p_u^F to 0.85 (for details see Section 6).

For a network of n nodes with $V = \{v_1, v_2, \cdots, v_n\}$, let $NR_i^{(t)} = NR^{(t)}(v_i)$ and denote the vector of node ranks at iteration t by $NR^{(t)} = (NR_1^{(t)}, NR_2^{(t)}, \cdots, NR_n^{(t)})^T$, where $t = 0, 1, \cdots$. We have

$$NR^{(t+1)} = \mathbf{T} \cdot NR^{(t)},$$

where $\mathbf{T}$ is a one-step transition matrix of the Markov chain defined by

$$\mathbf{T} = \begin{pmatrix} p^J_{11} & p^J_{12} & \cdots & p^J_{1n} \\ p^J_{21} & p^J_{22} & \cdots & p^J_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p^J_{n1} & p^J_{n2} & \cdots & p^J_{nn} \end{pmatrix} \cdot \begin{pmatrix} p^J_1 & 0 & \cdots & 0 \\ 0 & p^J_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p^J_n \end{pmatrix} + \begin{pmatrix} 0 & p^F_{12} & \cdots & p^F_{1n} \\ p^F_{21} & 0 & \cdots & p^F_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p^F_{n1} & p^F_{n2} & \cdots & 0 \end{pmatrix} \cdot \begin{pmatrix} p^F_1 & 0 & \cdots & 0 \\ 0 & p^F_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p^F_n \end{pmatrix} \quad (11)$$

Note that $\mathbf{T}$ is stable since it is a stochastic matrix having a maximum eigenvalue equal to one. This guarantees that the above recurrence relation converges to $NR^{(*)} = (NR_1^{(*)}, NR_2^{(*)}, \cdots, NR_n^{(*)})^T$, the steady state distribution [25]. This can be computed using a classic iterative scheme [23], given by Algorithm 1.

Algorithm 1 The NodeRank Computing Method

```
1: Given a positive value $\epsilon$, $i \leftarrow 0$
2: repeat
3:     $NR^{(i+1)} \leftarrow \mathbf{T} \cdot NR^{(i)}$
4:     $\delta \leftarrow \|NR^{(i+1)} - NR^{(i)}\|$
5:     $i++$
6: until $\delta < \epsilon$
```

5. MAPPING WITH NODERANK: RW-MAXMATCH AND RW-BFS ALGORITHMS

Taking advantage of the topology-aware node resource rank $NR^{(*)}$, we devise two new VN embedding algorithms called *RW-MaxMatch* and *RW-BFS*, both taking the substrate network and a virtual network as input. Given below are details of these two algorithms.

5.1 RW-MaxMatch

5.1.1 Description and Discussion

RW-MaxMatch is a two-stage VN embedding algorithm. During the first phase, it first computes the NodeRank value for each virtual node and substrate node. It then sorts virtual nodes in non-increasing order according to their NodeRank values, and does the same on substrate nodes. Let $m = |N_v|$ and $n = |N_s|$. Without loss of generality, let the sorted virtual nodes be

$$N_v^1, N_v^2, \cdots, N_v^m, \text{ where } NR_1 \geq NR_2 \geq \cdots \geq NR_m,$$

and let the sorted substrate nodes be

$$N_s^1, N_s^2, \cdots, N_s^n, \text{ where } NR_1 \geq NR_2 \geq \cdots \geq NR_n.$$

RW-MaxMatch maps N_v^1 to N_s^1, provided that the CPU capacity and the link capacity of N_s^1 can meet what node N_v^1 asks for. It then maps N_v^i to N_s^i in a similar way, where $i = 2, \cdots, m$. For convenience we call this mapping an *L2S2 mapping* (which stands for "large-to-large and small-to-small" mapping). This phase is executed in Algorithm 2.

In the second phase, RW-MaxMatch maps virtual links to substrate links (see Algorithm 3). The link embedding stage is similar to the previous research [28, 26]. That is, find the just-fit shortest paths by searching the k-shortest paths using binary search on values of k, until a path on the substrate network that satisfies the bandwidth requirement of the virtual link is found. If the substrate network supports path splitting, we will use the multi-commodity flow algorithm to embed the virtual links to the substrate links between the mapped virtual nodes.

Algorithm 2 RW-MaxMatch Node Mapping Stage

```
1: Compute the NodeRank values of all nodes in both $G_s$
   and $G_v$ using Algorithm 1 with a given value of $\epsilon$.
2: Sort the nodes in $G_s$ according to their NodeRank values
   in non-increasing order.
3: Sort the nodes in $G_v$ according to their NodeRank values
   in non-increasing order.
4: Map virtual nodes to substrate nodes using the L2S2
   mapping procedure.
5: if node resource constraints satisfied then
6:     return NODE_MAPPING_SUCCESS
7: else
8:     return NODE_MAPPING_FAILED
9: end if
```

Algorithm 3 RW-MaxMatch Link Mapping Stage

```
1: if path unsplittable then
2:     Map virtual links using the $k$-shortest path algorithm.
3: else
4:     Map virtual links using the multi-commodity flow al-
       gorithm.
5: end if
6: if link resource constraints satisfied then
7:     return LINK_MAPPING_SUCCESS
8: else
9:     return LINK_MAPPING_FAILED
10: end if
```

5.1.2 Time Complexity Analysis

Note that the iterative scheme (i.e., Algorithm 1) has been shown to yield any desired precision ϵ with a number of iterations proportional to $\max\{1, -log\epsilon\}$ [6]. The k-shortest path [12] link mapping algorithm and the multi-commodity flow problem [4] can both be solved in polynomial time. Thus, RW-MaxMatch is a polynomial-time algorithm in terms of $|G_s|$, $|G_v|$, and $\max\{1, -\log\epsilon\}$.

5.2 RW-BFS

5.2.1 Description and Discussion

The main framework of RW-BFS is presented in Algorithm 4 and the details of the function called $Match$ is presented in Algorithm 5. Different from RW-MaxMatch, RW-BFS is a one-stage backtracking VN embedding algorithm based on breadth-first search (BFS), which embeds virtual nodes and links at the same stage. The motivation of this algorithm is that a two-stage embedding algorithm maps all virtual nodes first without considering the impact on the link mapping stage, which may lead to unnecessary consumption of bandwidth resources of the substrate network. For example, when the virtual nodes in the virtual network are very close to each other (e.g., just one-hop away), the distance of two mapped virtual nodes in the substrate network may be considerably long. Thus, the virtual link between the two nodes would have to be mapped to a long path, resulting in a waste of the bandwidth resources.

To help solve this problem, RW-BFS first computes the NodeRank values of the nodes in G_s and G_v. It then constructs a breadth-first search tree of G_v, where the root node is the virtual node with the largest NodeRank value. At each level of the search tree, nodes are sorted by their NodeRank values in non-increasing order. For each virtual node N_v^i in G_v, we build a candidate substrate node list consisting of the nodes whose available CPU capacity and the sum of available bandwidth resources are at least as large as those of the virtual node. The substrate nodes in the list are also sorted by their NodeRank values in non-increasing order. In the last step, we embed every virtual node in G_v to the substrate node that meets the CPU resource requirement and maps the virtual links directly connected to a virtual node onto the substrate paths satisfying the bandwidth and connectivity constraints using the shortest path algorithm in a breadth-first search manner. If the virtual node is the root node of G_v, it is embedded to the substrate node with the largest NodeRank value. If there is no suitable mapping for N_v^i in its candidate substrate node list, we backtrack to the previous virtual node N_v^{i-1}, re-map it to another substrate node, and continue to map N_v^i. Notice that the constant value Max_Hop presented in the match function (Algorithm 5) is the maximum distance from the mapped parent node of N_v^i. The use of a hop bound is to reduce the search space of BFS. It can also avoid placing the virtual node too far away from the already mapped nodes to save substrate link resources.

Algorithm 4 RW-BFS

```
1: Compute the NodeRank values of all nodes in both $G_s$ and $G_v$ using Algorithm 1 with a given value of $\epsilon$.
2: Construct the breadth-first searching tree of $G_v$.
3: Sort all the nodes in $G_v$ in non-increasing order in each level of the breadth-first tree according to their NodeRank values.
4: backtrack_count = 0
5: Select a positive integer $\Delta$.
6: for each node $N_v^i$ in $G_v$ do
7:    Build the candidate substrate node list for $N_v^i$
8:    if Match($N_v^i$)==1 then
9:       Match($N_v^{i+1}$)
10:   else
11:      if backtrack_count <= $\Delta$ then
12:         Match($N_v^{i-1}$)
13:         backtrack_count++
14:      else
15:         return BFS_FAILED
16:      end if
17:   end if
18: end for
19: return BFS_SUCCESS
```

5.2.2 *Time Complexity Analysis*

It is well known that backtracking algorithm has an exponential time complexity. To avoid exponential explosion, we introduce an upper bound θ to limit the number of node remapping operations (in Algorithm 4 step 11).

Algorithm 5 The Details of Match Function

```
1: if $N_v^i$ is the root then
2:    map $N_v^i$ onto the substrate node with the largest NodeRank
3:    return MATCH_SUCCESS
4: end if
5: k = 1
6: while $k < Max\_Hop$ do
7:    for each $N_s^j$ which satisfies the k-hop constraint from its mapped parent node in the candidate substrate node list of $N_v^i$ do
8:       if pair($N_v^i$, $N_s^j$) meets all the capacity constraints then
9:          return MATCH_SUCCESS
10:      end if
11:   end for
12:   k++
13: end while
14: return MATCH_FAILED
```

6. PERFORMANCE EVALUATION

In this section we first describe the performance evaluation environment, and then present our main evaluation results. The experiments focus primarily on the performance comparison of the proposed two algorithms with the existing ones and the advantages of topology-aware node rank.

6.1 Evaluation Settings

We implement two simulators to evaluate the performance of our algorithms. The RW-MaxMatch simulator is a modified version of the VN embedding simulator devised in [3], and the RW-BFS simulator is implemented from scratch. Both simulators are publicly available at [2].

The substrate network topology is configured to have 100 nodes with about 500 links, corresponding to a medium sized ISP. We use the GT-ITM tool to generate substrate networks. The CPU and bandwidth resources of the substrate nodes and links are real numbers uniformly distributed between 50 and 100. We assume that VNRs arrive following a Poisson process with an average arrival rate of 5 VNs per 100 time units, and each one has an exponentially distributed lifetime with an average of 500 time units. In each VNR, the number of virtual nodes is determined by a uniform distribution between 2 and 20. The average VN connectivity is fixed at 50%, which means that an n-node VN has $n(n-1)/4$ links on average, similar with [28]. The CPU and bandwidth requirements of virtual nodes and links are real numbers uniformly distributed between 0 and 50. We run each of our simulations for about 50000 time units, which corresponds to about 2500 VNRs on average in one instance of simulation.

Our simulation experiments evaluate eight algorithms listed in Table 1. Ten different instances are run for these nine algorithms and we record the arithmetic mean of the ten runs as the final result. Supporting location constraints is not our focus in this paper, thus we don't compare our algorithms with algorithms proposed in [9], in which location constraints is taken into consideration. We have also considered to make a comparison between our algorithms and the virtual network embedding algorithm based on subgraph isomorphism proposed in [21]. However, the algorithm in [21] is limited to substrate networks and virtual networks

Table 1: Comparisons

Notation	Algorithm Description
RW-MM-SP	Use the RW model to compute NodeRank values. Two-stage mapping: first map virtual nodes with larger NodeRank values to the substrate nodes with larger NodeRank values, then embed the virtual links using shortest path if the substrate does not support path splitting. MM denotes MaxMatch.
CB-MM-SP	Simply use Equation (6) to compute node resources rank. The other steps of CB-MM-SP is the same as RW-MM-SP. CB denotes multiplying CPU by bandwidth.
RW-BFS	Use the RW model to compute NodeRank values. One-stage mapping: embed the virtual nodes and links during the same stage based on breadth-first searching.
CB-BFS	Simply use Equation (6) to compute node resources rank. The other steps of CB-BFS is the same as RW-BFS.
RW-MM-MCF	Similar to RW-MM-SP, with path splitting.
CB-MM-MCF	Similar to CB-MM-SP, with path splitting.
BL-SP	The baseline algorithm proposed in [28].
BL-MCF	The baseline algorithm using MCF to embed virtual links.

modeled by directed graphs, thus a fair comparison is not applicable and so we exclude it from the results.

The bias factors of p_v^J and p_v^F presented in Equation (10) are used to balance between local and global node resources. Experiments show that setting p_v^J to 0.15 and p_v^F to 0.85, the same values used in PageRank, will achieve the optimal result. The value for the parameter ϵ presented in the iterative scheme is set to 0.0001. Since the diameter of the substrate network is 3, we set the value of Max_Hop in the match function (Algorithm 5) to 3. To gain better mapping quality without consuming much execution time, the upper bound Δ, which limits the re-mapping steps in RW-BFS, is set to $3n$, where n is the number of virtual nodes in the VNR.

6.2 Evaluation Results

Our evaluation results quantify the efficiency of the two algorithms we proposed (depicted in Fig. 3(a), Fig. 3(b), and Fig. 3(c)) and reveal the benefits of using the RW model for computing the node resource rank (depicted in Fig. 4(a), Fig. 4(b), and Fig. 4(c)). Several performance metrics for evaluation purposes are used, including the long-term average revenue defined by Equation (3), the VNR acceptance ratio defined by Equation (4), the long term R/C ratio defined by Equation (5), and the runtime of these algorithms. We summarize the key observations from our simulation as follows.

6.2.1 *Our new algorithms lead to higher acceptance ratio and average revenue*

Fig. 3(a) and Fig. 3(b) show that the proposed algorithms RW-MM-SP and RW-BFS can produce higher revenue and acceptance ratio than those of the baseline algorithms, regardless the path splittability of substrate network. These two graphs show that topology-aware node ranking plays an important role in VN embedding process. We note that although RW-BFS produces the highest acceptance ratio, the long-term average revenue of this algorithm is lower than that of RW-MM-MCF. The reason is that RW-BFS tends to accept more small VNRs and reject a number of large VNRs, thus the residual resources on the substrate network would have larger fragmentation.

6.2.2 *RW-BFS produces the hightest long-term R/C ratio*

The reason is that RW-BFS uses breadth-first search to map virtual nodes, and avoids mapping virtual links onto a long substrate path that may cause more resource consumption. Fig. 3(a) shows that, in the case of un-splittable paths, RW-BFS produces the highest revenue, which has close relationship with its high long-term R/C ratio (depicted in Fig. 3(c)). Lower VN embedding cost saves more room for the future VNRs.

6.2.3 *NodeRank is a better node resource measure*

Computing NodeRank leads to higher revenue and acceptance ratio than simply using Equation (6) as the measure of node resource. With respect to the two-stage VN embedding algorithms, Fig. 4(a) and Fig. 4(b) show that RW-MM-SP and RW-MM-MCF offer larger revenue and better acceptance ratio than CB-MM-SP and CB-MM-MCF. The NodeRank also contributes to the increase of revenue and acceptance ratio for one-stage VN embedding algorithms. With NodeRank, the resource ranking is not only determined by the node itself, but also influenced by its neighbors. A node with higher NodeRank value tends to have a set of good neighbors with higher NodeRank values, which may increase probability of meeting the constraints of the VN request and thus benefit the VN embedding process. However, shown in Fig. 4(c), NodeRank makes no significant contributions to increasing R/C ratio.

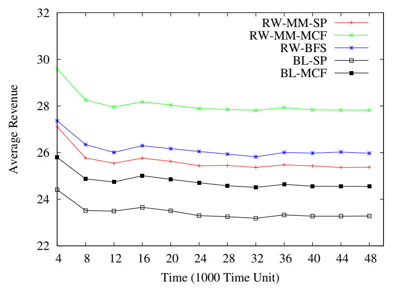
6.2.4 *The RW-based algorithms consume comparable runtime with others*

Fig. 5 depicts the average execution time of these VN embedding algorithms run on the same PC. The RW-based algorithms consume nearly the same time as the CB-based algorithms, even though the RM-based algorithms need to compute node ranks, both in the substrate network and virtual network, for each VNR. The iterative process of computing the node rank converges to a reasonable tolerance in roughly 7 iterations for a substrate network with 100 nodes and about 500 links, which is deemed reasonable.

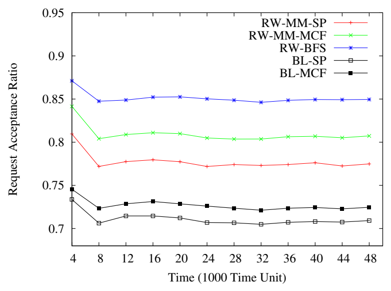
7. CONCLUSION

In this paper, we compute the topology-aware node resource rank based on Markov Random Walks. Using the resource rank of a node as its resource measurement, we propose two new algorithms called RW-MaxMatch and RW-BFS. Simulation results show that our algorithms outperform the previous approaches in terms of the long-term average revenue and the VNR acceptance ratio. We also demonstrate the benefit of applying the RW model to compute the node resource rank for achieving a higher acceptance ratio and higher revenue.

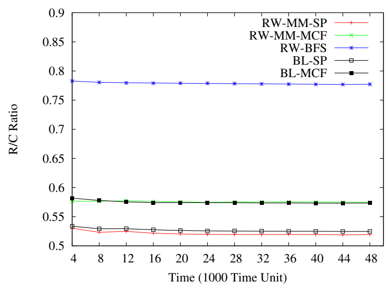
In this work, we define the amount of node resources presented in Equation (6) the same as the previous work in [28].



(a) The long-term average revenue comparison

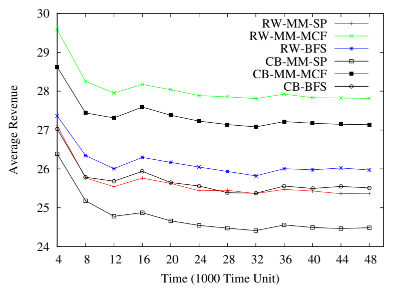


(b) The VNR acceptance ratio over time comparison

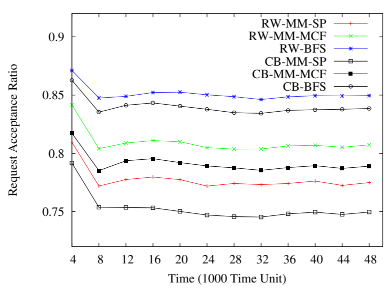


(c) The long-term R/C ratio (Revenue/Cost) comparison

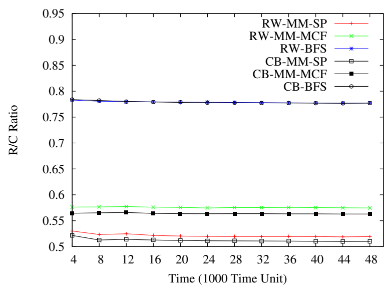
Figure 3: Comparison between the RW-based and the existing algorithms



(a) The long-term average revenue comparison



(b) The VNR acceptance ratio over time comparison



(c) The long-term R/C ratio (Revenue/Cost) comparison

Figure 4: Comparison between the RW-based and CB-based algorithms

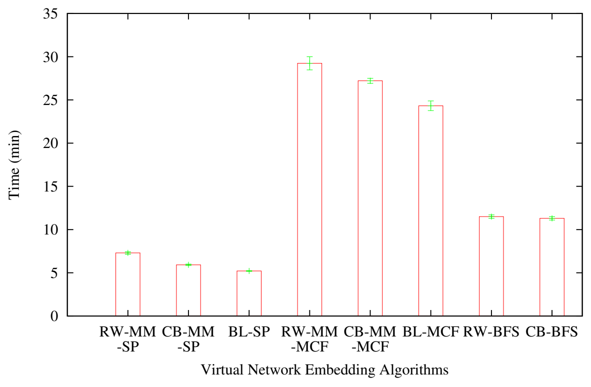


Figure 5: The execution time comparison of embedding algorithms with 95% confidence interval

But we also find using other methods to mark the node resources value, such as α·CPU + $(1-\alpha)$·BW $(0 < \alpha < 1)$, appears to get better results in our preliminary experiments. The value of α can be changed dynamically to adapt to the substrate network resource environment when mapping a new virtual network request, and such study is left for further development. Secondly, the different value of bias expressing the probability whether a node choosing to jump to a random node or following an edge from this node will also be tested in our future work. Finally, we plan to extend our work by considering the domain-specific substrate resource availability (e.g., fat-tree type network topology) and requirements from user requests (e.g., multi-dimensional resource requirements on CPU, storage, etc.).

Acknowledgment

This work is supported in part by the 973 project of China (2009CB320504), Funds for Creative Research Groups of China (60821001), NSF CNS-0709001, NSF/BBN GENI subaward, NSF CCF-0830314 and NSF CNS-1018422.

8. REFERENCES

[1] Amazonec2,http://aws.amazon.com/ec2/.

[2] VNE-RW Simulator http://int.bupt.edu.cn/~sensu/vne-rw.html.

[3] VNE Simulator http://www.cs.princeton.edu/~minlanyu/embed.tar.gz.

[4] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, and K. Weihe. *Network flows: theory, algorithms, and applications.* Prentice hall Englewood Cliffs, NJ, 1993.

[5] David G. Andersen. Theoretical approaches to node assignment. Unpublished Manuscript, December 2002.

[6] M. Bianchini, M. Gori, and F. Scarselli. Inside pagerank. *ACM Transactions on Internet Technology (TOIT)*, 5(1):92–128, 2005.

[7] N. Butt, M. Chowdhury, and R. Boutaba. Topology-Awareness and Re-optimization Mechanism for Virtual Network Embedding. In *Networking 2010: 9th International Ifip Tc 6 Networking Conference, Chennai, India, May 11-15, 2010, Proceedings*, page 27. Not Avail, 2010.

[8] N.M.M.K. Chowdhury and R. Boutaba. Network virtualization: state of the art and research challenges. *IEEE Communications magazine*, 47(7):20–26, 2009.

[9] N.M.M.K. Chowdhury, M.R. Rahman, and R. Boutaba. Virtual network embedding with coordinated node and link mapping. In *IEEE INFOCOM*, 2009.

[10] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento. An improved algorithm for matching large graphs. In *3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, pages 149–159, 2001.

[11] M. Diligenti, M. Gori, M. Maggini, and D. di Ingegneria dell'Informazione. A unified probabilistic

framework for web page scoring systems. *IEEE Transactions on knowledge and data engineering*, 16(1):4–16, 2004.

[12] D. Eppstein. Finding the k shortest paths. In *Proc. of IEEE Symposium on Foundations of Computer Science.* IEEE Comput. Soc. Press, 1994.

[13] J. Fan and M. Ammar. Dynamic topology configuration in service overlay networks: A study of reconfiguration policies. In *Proc. IEEE INFOCOM*, 2006.

[14] N. Feamster, L. Gao, and J. Rexford. How to lease the Internet in your spare time. *ACM SIGCOMM Computer Communication Review*, 37(1):64, 2007.

[15] Global Environment for Network Innovations. National Science Foundation, http://www.geni.net/, August 2005.

[16] M. Gori, M. Maggini, and L. Sarti. Exact and approximate graph matching using random walks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1100–1111, 2005.

[17] C. Guo, G. Lu, H.J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, Y. Zhang, MSR Asia, and MSR Redmond. SecondNet: A Data Center Network Virtualization Architecture with Bandwidth Guarantees.

[18] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener. Provisioning a virtual private network: A network design problem for multicommodity flow. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 389–398. ACM New York, NY, USA, 2001.

[19] J. He, R. Zhang-Shen, Y. Li, C.Y. Lee, J. Rexford, and M. Chiang. Davinci: Dynamically adaptive virtual networks for a customized internet. In *Proceedings of the 2008 ACM CoNEXT Conference*, pages 1–12. ACM, 2008.

[20] I. Houidi, W. Louati, and D. Zeghlache. A distributed virtual network mapping algorithm. In *Proceedings of IEEE ICC*, pages 5634–5640, 2008.

[21] J. Lischka and H. Karl. A virtual network mapping algorithm based on subgraph isomorphism detection. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pages 81–88. ACM, 2009.

[22] J. Lu and J. Turner. Efficient mapping of virtual networks onto a shared substrate. *Department of Computer Science and Engineering, Washington University in St. Louis, Technical Report WUCSE-2006*, 35, 2006.

[23] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. *Technical report, Stanford Digital Library Technologies Project*, 1998.

[24] R. Ricci, C. Alfeld, and J. Lepreau. A solver for the network testbed mapping problem. *ACM SIGCOMM Computer Communication Review*, 33(2):81, 2003.

[25] E. Seneta. *Non-negative matrices and Markov chains.* Springer Verlag, 2006.

[26] W. Szeto, Y. Iraqi, and R. Boutaba. A multi-commodity flow based approach to virtual network resource allocation. In *Proc. GLOBECOM: IEEE Global Telecommunications Conference*, 2003.

[27] JS Turner and DE Taylor. Diversifying the internet. In *IEEE Global Telecommunications Conference, 2005. GLOBECOM'05*, volume 2.

[28] M. Yu, Y. Yi, J. Rexford, M. Chiang, et al. Rethinking virtual network embedding: Substrate support for path splitting and migration. *COMPUTER COMMUNICATION REVIEW*, 38(2):17, 2008.

[29] Y. Zhu and M. Ammar. Algorithms for assigning substrate network resources to virtual network components. In *Proc. IEEE INFOCOM*, 2006.