A Latency-Aware Reward Model Based Greedy Heuristic for the Virtual Network Embedding Problem

Francesco Bianchi
Department of Civil Engineering and Computer Science Engineering
University of Rome "Tor Vergata", Italy
f.bianchi@ing.uniroma2.it

Francesco Lo Presti
Department of Civil Engineering and Computer Science Engineering
University of Rome "Tor Vergata", Italy
lopresti@info.uniroma2.it

ABSTRACT

The increasing use of virtualization (e.g., in Cloud Computing, Software Defined Networks), demands to Infrastructure Providers (InPs) to optimize the placement of the virtual network requests (VNRs) into a substrate network. In addition to that, they need to cope with QoS, in particular for the rising number of time critical applications (e.g., healthcare, VoIP). Granting resource optimization along with QoS compliance, are two competing goals. In this work, we propose a two-stage virtual network embedding (VNE) algorithm, which maps first virtual nodes to substrate nodes based on a suitable latency-aware ranking algorithm and then maps links along the shortest paths, in terms of latencies. The central component of our approach is a new node ranking algorithm, MCRR-LA, based on Markov Reward Processes, which associates a metric that accounts for and well captures the amount of local resources combined with latency values available in the vicinity of a certain node. The metric is complemented with a Breadth-First search. We widely evaluated our algorithm through simulation. Our experiments point out that our algorithm is able to reduce the average path delay while granting good resource performances in terms of lower VNR blocking rate and higher revenues. We compared our algorithm with a previous two-stage approach obtaining good results useful to underline the strengths of the novel approach.

Keywords

Cloud computing; delay; latency; Markov chain; Markov reward process (MRP); network virtualization; node ranking; quality of service (QoS); random walk; topology-aware; virtual network embedding (VNE)

1. INTRODUCTION

The base for the future Internet is characterized by the Infrastructure as a Service (IaaS) service model [11], producing a decoupling of the ISP (Internet Service Provider) into two novel players: the Infrastructure Provider (InP) and the Service Provider (SP). The InPs (e.g., Cloud providers), who own and manage the infrastructure, rent out parts of the network resources to the SPs (e.g., Cloud users), who in turn meet the end-users demands, making virtual networks (VNs) and providing end-to-end services [10, 11]. In this very adaptive multi-layer architecture, the InPs have the challenging task to manage the substrate network efficiently in order to maximize the number of mapped virtual network requests (VNRs) along with the revenue. In practice, the InPs have to determine, on-line, a subset of physical nodes and links with sufficient resources to host each VNR composed by a set of virtual nodes and links with a certain amount of resource attributes, e.g., computational capacity, link bandwidth.

This problem is referred to as Virtual Network Embedding (VNE) which is a well known NP-complete problem [1, 22]. In order to support on-line operations, many heuristics have been proposed in the literature (e.g., [24, 17, 22, 9, 8, 23, 20, 16]) under different settings and assumptions.

In this paper, we study the VNE embedding problem under maximum latency QoS constraints. Our goal is to determine a set of node and link resources in a substrate network which minimize the end-to-end latency between pair of nodes. This is motivated by the growing number of multimedia and other delay-sensitive applications (Cisco envisioned that about 90% of Internet traffic is related to delay-sensitive applications [14]).

Building on our prior work on VNE embedding [6], we present a new algorithm, called MCRR-LA (Markov Chain with Rewards Ranking-Latency Aware), for the VNE problem subjected to QoS constraints (e.g., latency). The approach is inspired by [13, 8, 23] which devised so called *coordinated* two-stage VNE algorithms: in the first stage the algorithm embeds virtual nodes into physical (also called substrate) nodes; then, in the second stage, the algorithm embeds the logical links into physical paths between the nodes. To this end, they adopt a suitable metric which captures the availability of resources, in a node neighborhood. The mapping is then performed adopting a simple greedy strategy, by ranking the virtual and physical nodes utilizing the proposed metric. These works draw inspiration from the PageRank algorithm [7] which is extensively employed to rank web pages, and the contrived metrics imitate the PageRank concept by associating to nodes a metric which represents their relative "importance" in terms of resources.

Following the two-stage approach idea, we propose a novel latency-aware metric for a two-stage coordinated VNE al-

VALUETOOLS 2016, October 25-28, Taormina, Italy

DOI 10.4108/eai.25-10-2016.2266742

gorithm: MCRR-LA. Differently from earlier methods, our ranking metric relies on the accumulated reward of a suitable Markov Chain. The intuition is that the accumulated reward, with each node reward being function of the amount of the node resources, penalized proportionally according to the delay over the physical links, should well capture the amount of resources available in a node area and the overall QoS (in term of latency). In this way, this combined metric is useful also to detect low-latency network areas. In the first phase, both virtual and substrate nodes are ranked based on the latency-aware metric. Then, substrate nodes are rearranged in a new list which holds in the first positions those nodes which not only have more resources but that also are closer to the highest ranked substrate node. Subsequently, virtual nodes are mapped, in a greedy manner, the highest ranked virtual node to the first substrate node in the list, the second highest virtual node to the second substrate node, etc.. In the second phase the algorithm embeds each virtual link onto a physical path, using the shortest path between each pair of originally connected nodes. We have evaluated our algorithm through simulation. Our results show that our solution is able to greatly reduce the average path delay of the embedded VNRs almost preserving good performances regarding blocking probability and revenue.

The rest of the paper is organized as follows. In section 3 we formalize the VNE problem. The new ranking algorithm (MCRR-LA), based on Markov Reward Processes, and the VNE algorithm are presented in section 4. In section 5 we evaluate MCRR-LA through simulation. Section 6 concludes the paper.

2. RELATED WORK

The VNE is a well known widely investigated problem which has been studied under different assumptions. Since the VNE problem is NP-complete [1, 22] most solutions focus on efficient heuristics (e.g., [24, 17, 22, 9, 8, 23, 20, 16]).

Among the others, many algorithms work in two successive stages for the node and for the link mapping. Lately, a new class of two-stage VNE algorithms has been proposed, named coordinated algorithms (e.g.: CO-VNE [13]). The principal feature of this new category of algorithms (e.g., [13, 8, 23]) is that the information concerning the links, required for the second stage (i.e., link mapping), is also considered in the first one (i.e., node mapping). In particular, these solutions take into account the topology, by associating to each node a metric which represents the resources of the node and of those of its neighbours, combined in one metric. The embedding is then executed via an easy greedy approach, by ranking the virtual and substrate nodes according to this metric and mapping the highly ranked virtual nodes onto the matching highly ranked substrate ones. Noticeably, the majority of these works drew inspiration from the PageRank algorithm [7]. The PageRank algorithm relies on the assumption that the importance of a web page is function of the importance of the web pages that link to it: the higher the number of incoming links and the importance of the web pages that link to it, the more important the web page is. By interpreting the web surfing as a Random Walk, it turns out that the importance of a web page corresponds to the stationary probability of a Markov Chain induced by the web links themselves, where the pages represent the Markov Chain states and the links the transitions between states. Following the same approach, the aforementioned VNE algorithms associate to each node a metric which captures the relative "importance" of a node in terms of resources, which accounts for the local resources and those of the connected links and the nearby nodes. As for the PageRank algorithm, this metric can be computed as a stationary probability of a suitable Markov Chain.

As explained in the introduction, while we consider a similar approach, we follow a totally different direction in determining the ranking metrics and adopt the cumulated reward of a suitable Markov Chain rather than the node stationary probability.

Many QoS objectives have been taken into account in solving the VNE problem: energy efficiency [18], survivability and resiliency [19], etc. [11]. Nevertheless, just a few researchers studied the latency-aware VNE. Chowdhury et al. [9] consider a requested location constraint (e.g., using geographic coordinates) for every virtual node and a maximum distance threshold indicating how far from the demanded location can be placed each node. They solve an optimization problem using mixed integer programming. This type of approach deals indirectly with delay.

Similar to our approach, Behrouznia et al. in a recent work [4] propose a two-stage QoS-aware algorithm, where the key feature is the adapted computation of quality of the physical paths, introduced in [21]. The objective metrics are both cost-effectiveness and QoS. In the first phase, the algorithm sorts the substrate nodes according to the metric of the node's available resources (available computational capacity of the node multiplied by the outgoing available bandwidth). In the second phase, in order to map each virtual link, the algorithm finds the K shortest paths between each couple of involved mapped virtual nodes (through K-Shortest-Paths algorithm). Then it computes the quality of the links making up the paths that comply with the requested bandwidth and QoS constraints of the relative virtual link. The virtual link is then mapped on the highest quality path. Differently, we calculate instead a topology-aware metric for both the request and substrate network. For the substrate network the metric is also delay-aware and is supplemented by a mechanism to shorten the physical paths.

3. VNE: PROBLEM DESCRIPTION

3.1 Substrate Network

We model the substrate network as an undirected graph, weighted on both nodes and edges, $G^s = (N^s, E^s, C^s, B^s, L^s)$, where N^s is the set of substrate nodes, E^s is the set of substrate links, C^s is the set of available CPU capacity associated with each node[1], B^s is the set of available bandwidth associated with each link and L^s is the set of the network delay associated to each link. We denote the set of substrate paths, without cycles, with P^s. Figure 1 reports an example of substrate network where the numbers inside the rectangles denote the available node CPU capacities, those near the edges the link available bandwidth, and the numbers inside green hexagons the link latency.

3.2 Virtual Network Request

[1]We only consider the computational capacity as node resource metric. The algorithm can be extended in case of multiple resources, *e.g.*, memory, I/O bandwidth

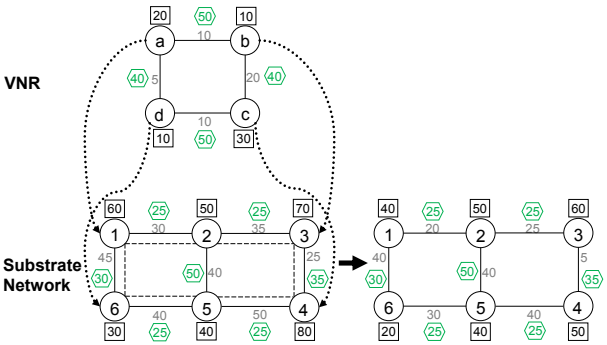


Figure 1: A VNR, the substrate network and a feasible mapping of the VNR with the deriving updated substrate network. Numbers inside rectangles near the nodes (circles), those near the edges (solid lines) and those inside green hexagons, represent the computational capacity, the link bandwidth and the link latency, respectively. Dotted lines and dashed lines show the node mapping and the consequent link mapping, respectively.

Table 1: Notations

Notations	Meaning
G^s	Substrate network
N^s	Set of substrate nodes
E^s	Set of substrate links
C^s	Set of available CPU capacities
B^s	Set of available bandwidth capacities
L^s	Set of latencies
P^s	Set of substrate paths without cycles
G^v	Virtual network request
N^v	Set of virtual nodes
E^v	Set of virtual links
C^v	Set of CPU requests
B^v	Set of bandwidth requests
L^v	Set of latency constraints
t_a	Arrival time of the VNR
t_d	Lifetime of the VNR

We also model a virtual network request as an undirected graph, weighted on both nodes and edges, $G^v = (N^v, E^v, C^v, B^v, L^v)$, where N^v is the set of virtual nodes, E^v is the set of virtual links, C^v is the set of requested CPU capacity associated with each node, B^v is the set of requested bandwidth associated with each link and L^v is the set of the network delay constraints associated to each link. We will also write $VNR(t_a, t_d)$ to denote a VNR with arrival time t_a and lifetime t_d. Figure 1 shows a VNR made up of four nodes and four links with the same notation of the substrate network about the requested capacities and latencies. The set of notations is reported in Table 1.

3.3 Virtual Network Mapping

Virtual network mapping $M : G^v \to G^s$ is the embedding of a VNR, G^v, in the physical network G^s. In particular, the set of virtual nodes is embedded into a subset of substrate nodes and the set of virtual links is embedded into a subset of substrate links, namely a subset of the substrate paths P^s. This procedure is made up of two phases: node mapping and link mapping.

The node mapping phase provides a mapping function from a virtual node to a substrate one, $M_{node} : N^v \to N^s$, such that: $\forall n^v, m^v \in N^v$

$$M_{node}(n^v) = n^s \in N^s, \tag{1}$$

subject to:

- $M_{node}(n^v) = M_{node}(m^v)$ if and only if $m^v = n^v$;
- $C^v(n^v) \leq C^s(M_{node}(n^v))$.

Each virtual node is mapped to a distinct physical node provided that the latter has at least the same quantity of available resources. In Figure 1 the resulting node mapping for the VNR is $\{a \to 1, b \to 3, c \to 4, d \to 6\}$, graphically represented with dotted lines.

The link mapping provides a mapping function of a virtual link to a substrate path, with no cycles, consisting of one or more links, $M_{link} : E^v \to P^s$, such that: $\forall e^v = (m^v n^v) \in E^v$

$$M_{link}(m^v n^v) = p^s(M_{node}(m^v), M_{node}(n^v)) \in P^s(M_{node}(m^v), M_{node}(n^v)),$$

subject to:

- $B^v(e^v) \leq \min_{e^s \in p^s(M_{link}(e^v))} B^s(e^s)$;
- $L^v(e^v) \geq \sum_{e^s \in p^s(M_{link}(e^v))} L^s(e^s)$.

Each substrate link in the path, must possess at least the same quantity of resources of the virtual link. Furthermore, the overall latency of each physical path must comply with the requested virtual link's latency constraint. In Figure 1 the resulting link mapping for the VNR is $\{(ab) \to (12, 23), (bc) \to (34), (cd) \to (45, 56), (da) \to (61)\}$, denoted by dashed lines.

3.4 Objective

The goal of the InP is to maximize the revenue, while optimizing resource utilization. The SP, instead, wants to pay just the agreed resource costs. Following previous approaches [22, 24, 9, 23, 13], we define the revenue $R(G^v, t_a, t_d)$ for the InP for accepting a VNR with lifetime t_d, at time t_a, as:

$$R(G^v, t_a, t_d) = \begin{cases} R_0(G^v, t_a, t_d) \cdot t_d, & \text{if accepted} \\ 0, & \text{otherwise} \end{cases}, \tag{2}$$

where $R_0(G^v, t_a, t_d)$ represents the revenue per time unit for $VNR(t_a, t_d)$ that can be defined as:

$$R_0(G^v, t_a, t_d) = \alpha_c \sum_{n^v \in N^v} C^v(n^v) + \alpha_b \sum_{e^v \in E^v} B^v(e^v), \tag{3}$$

where α_c is the unit price for the computational resources and α_b is the unit price for the bandwidth resources.

Generally, as highlighted in other works [22, 9, 8], the main objective for the InP is to maximize the long term time-average revenue:

$$R_{tot} = \lim_{T \to \infty} \frac{\sum_{i=1}^{N_T} R(G^{v,(i)}, t_a^{(i)}, t_d^{(i)})}{T}, \tag{4}$$

where N_T is the number of requests arrived within time T, and $R(G^{v,(i)}, t_a^{(i)}, t_d^{(i)})$ is the revenue gained from the i-th VNR with arrival time $t_a^{(i)}$ and lifetime $t_d^{(i)}$.

An additional important index, which has an impact on the former one, is the blocking ratio, namely the blocking probability:

$$BR = \lim_{T\to\infty} \frac{\sum_{j=1}^{NREJ_T} VNR_{rej}^{(j)}(t_a^{(j)}, t_d^{(j)})}{\sum_{i=1}^{N_T} VNR^{(i)}(t_a^{(i)}, t_d^{(i)})}, \quad (5)$$

where $NREJ_T$ is the total number of rejected VNRs within time T and $VNR_{rej}^{(j)}(t_a^{(j)}, t_d^{(j)})$ is the j-th rejected VNR.

Furthermore, as suggested in [2], we make use of the average path delay of each accepted VNR as a key metric which meaning is how well a VNR performs after it has been mapped into the physical network.

4. MARKOV CHAIN REWARDS BASED LATENCY AWARE NODE RANKING ALGORITHM

Following previous works in the literature [8, 23, 13], our VNE algorithm is made up of two distinct, sequential stages, namely, the node mapping stage and the link mapping stage: when a VNR arrives, first of all virtual nodes are mapped to physical nodes with adequate resources following a greedy strategy which also takes into account the link latency; then, in the link mapping stage, virtual links are mapped to substrate paths with sufficient link capacity, following a shortest-path (in terms of latency values) based link mapping.

In section 4.1, we first present our approach to latency aware node ranking, MCRR-LA, based on Markov Chains with rewards model [12], which captures the amount of resources (computational capacity, network bandwidth) available in a node and its surrounding area coupled with the network latency between nodes. Then, in Section 4.2 we describe the node and link mapping algorithms in detail.

4.1 MCRR-LA Node Ranking Metric

As noticed in [13], a ranking metric for the VNE problem should account for both local and neighboring nodes resources. By intuition, the higher the amount of resources in terms of computing and network capacity in a node neighborhood, the more likely a virtual network can be embedded in that portion of the physical network. We are going to add to the above mentioned criteria a penalization proportional with respect to the level of latency.

First of all, we introduce the concept of latency-aware node resources. Without loss of generality, we follow an approach similar to the one proposed by Zhang et al. in [23][2] and formulate the amount of available resources $CBL(n)$ of a node n by the product of the available CPU resource and the sum of the bandwidth resources, each multiplied by the normalized link latency as follows:

$$CBL(n) = C^s(n) \cdot \sum_{m\in N(n)} \left[B^s((n,m)) \cdot \left(1 - \left(\frac{L^s((n,m)) - min\_lat}{max\_lat - min\_lat}\right)\right)\right], \quad (6)$$

[2]The ranking metric can be similarly defined using alternative resource metrics, *e.g.*, [13].

where $N(n)$ is the set of neighbors of node n. In (6), $L^s((n,m))$ is the latency over the physical link between nodes n and m, and $[min\_lat, max\_lat]$ is the range of latency values in the network. The latency in CBL is thus normalized in the interval $[0,1]$, with 1 corresponding to links with minimum latency and 0 to links with maximum latency. The node resources are then normalized as follows:

$$Res(n) = \frac{CBL(n)}{\sum_{m\in N^s} CBL(m)}. \quad (7)$$

We define the MCRR-LA ranking metric $V_\gamma(n)$ of node n, recursively, as a weighted sum of the local resources and the neighbouring nodes metric as follows

$$V_\gamma(n) = (1-\gamma)Res(n) + \gamma \sum_{m\in N(n)} \frac{Res(m)}{\sum_{h\in N(n)} Res(h)} V_\gamma(m), \quad (8)$$

with $\gamma \in [0,1)$ the relative weight of the neighbours metric. In (8), each neighbour contributes to the sum in proportion to its amount of resources with respect to the other adjacent nodes (represented by the factor $\frac{Res(m)}{\sum_{h\in N(n)} Res(h)}$). To rewrite (8) in a compact form, let $\mathbf{P}$ a $|N| \times |N|$ matrix defined as follows:

$$P(n,m) = \begin{cases} \frac{Res(m)}{\sum_{h\in N(n)} Res(h)} & \text{if } (n,m) \in E^s \\ 0 & \text{otherwise} \end{cases}, \quad (9)$$

By construction, P is a stochastic matrix with all rows summing to 1. We can now conveniently rewrite (8) as:

$$V_\gamma(n) = (1-\gamma)Res(n) + \gamma \cdot \sum_{n'\in N} P(n,n')V_\gamma(n'), \quad (10)$$

or in matrix form

$$\mathbf{V}_\gamma = (1-\gamma)\mathbf{Res} + \gamma \mathbf{P}\mathbf{V}_\gamma, \quad (11)$$

where $\mathbf{Res} = (Res(1), Res(2), \ldots, Res(|N|))^T$, and $\mathbf{V}_\gamma = (V_\gamma(1), V_\gamma(2), \ldots, V_\gamma(|N|))^T$.

The recursive equations (11) have an elegant physical interpretation. Indeed, (11) can be regarded as the Bellman equations [5] of the discounted cumulative reward, with discount factor γ, of the Markov Chain with transition probability matrix $\mathbf{P}$ over the set of nodes N, and rewards vector $\mathbf{Rew} = (Rew(1), Rew(2), \ldots, Rew(|N|))$, with $Rew(n) = (1-\gamma)Res(n)$, $n \in N$. In other words, the ranking metric of node n, $V_\gamma(n)$, is the expected discounted accumulated reward of a Markov Chain with transition probability $\mathbf{P}$, that is

$$V_\gamma(n) = \lim_{k\to\infty} \mathbb{E}_{\mathbf{P}} \left[\sum_{i=0}^{k} \gamma^i Rew(n_i) \right], \quad (12)$$

where $n_0, n_1, n_2, \ldots,$ denotes a sample path with initial state $n_0 = n$.

$\mathbf{V}_\gamma$ can be calculated as the unique solution of Eq.(11). By observing that since $\mathbf{P}$ is stochastic, $(\mathbf{I} - \gamma\mathbf{P})$, $0 \le \gamma < 1$, is invertible, and we readily obtain

$$\mathbf{V}_\gamma = (\mathbf{I} - \gamma\mathbf{P})^{-1}(1-\gamma)\mathbf{Res}. \quad (13)$$

From the above definition, it is clear that the higher the node ranking $V_\gamma(n)$ of a node, the higher resource in the node and its neighborhood. In addition, if we consider nodes with same resource state in their respective areas, we will experience lower $V_\gamma(n)$ values for the nodes with low latency

neighbourhood, since higher latency values affect more negatively the (accumulated) rewards. The discount factor γ is a measure of the size of the neighborhood taken into account to determine the node metric: $\gamma = 0$ only the local resources are taken into account, while as γ increases, larger and larger portion of the graph close to a node is accounted for in the metric. It is worth noting that while metric $V_\gamma(n)$ has been so far implicitly applied only to the substrate network nodes, it can be applied to the virtual network nodes too. In this latter case, though, instead of the amount of available resources, $V_\gamma(n)$ represents the amount of resources *required* by a virtual node and its neighboring nodes. This demand-availability relationship suggests that as a heuristic we can simply map virtual nodes with higher ranking metric to physical nodes with higher ranking.

4.2 Full Mapping Algorithm

As we mentioned before, our approach to solve the VNE problem consists of a coordinated two-stage algorithm. In the first phase, namely nodes mapping (Algorithm 1), we first order them in non increasing order according to the MCRR-LA metric (line 12-13). Then, using the physical node with the highest metric value (line 14) as root node, we determine a subset of nodes using a Breadth-First search (BFS) (lines 17-23). The starting maximum search hop-distance is obtained dividing the number of virtual nodes by the average degree of the substrate nodes (line 16). In this BFS search, only the nodes with computing capacity greater than or equal to the minimum computing capacity required by the VNR nodes are considered. Unless the BFS finds at least the sufficient number of physical nodes to host the requested nodes, the search is extended by increasing the searching distance by one unit (line 20-21). Next, two lists are created, both sorted according to the MCRR-LA metric (line 24-26): the first, containing the substrate nodes selected by the BFS search; the second, with the rest of the substrate nodes. The two lists are then concatenated. We then proceed to the node mapping itself (lines 28-36): using a simple greedy approach, following the order determined by the MCRR-LA metric, the virtual nodes are mapped to the first physical node in the above list which has adequate resources to host them, provided that each physical node can just host one virtual node. If all virtual nodes are successfully embedded, the VNE mapping continues with the second phase, otherwise, the mapping procedure aborts and the VNR is blocked.

In the second phase (Algorithm 2), the algorithm computes the shortest path (Dijkstra algorithm), in terms of latency, between each pair of substrate nodes. For the sake of efficiency, the substrate links without enough bandwidth are cut before executing the calculation of the shortest paths (lines 6-10). If it is impossible to find a path between two physical nodes, meeting both bandwidth and latency constraints demanded by the currently processed virtual link, the heuristic is unable to determine a suitable assignment and the VNR is refused (lines 14-22).

Remark The second stage simply determines the shortest path among nodes, thus selecting the minimum latency path among them, irrespectively of the number of hops among nodes. In order to minimize resource consumption, the algorithm should also minimize the number of hops. We achieve this goal implicitly by using the BFS algorithm. Indeed, using the BFS search in the first stage we determine as best candidates for node mapping not only the nodes with higher resources in their vicinity, but also the nodes which are closer in terms of network hops (using the best node - the root - as reference node).

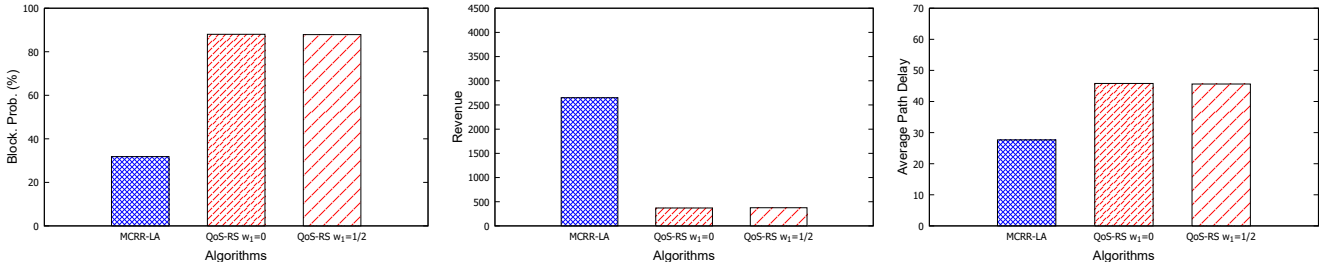
Algorithm 1 *NodeMapping*

Input : VNR $G^v(N^v, E^v, C^v, B^v, L^v)$, Substrate net $G^s(N^s, E^s, C^s, B^s, L^s)$, nodes rank vector for VNR V^v_γ, nodes rank vector for substrate net V^s_γ ;
Output: Mapping of nodes M_{node};

1: data initialization;
2: $M_{node} \leftarrow \mathbf{0}$;
3: $counter \leftarrow 0$;
4: $nodes\_mapping\_success \leftarrow false$;
5: $root\_node \leftarrow \mathbf{0}$;
6: $search\_dist \leftarrow 0$;
7: $average\_degree \leftarrow 0$;
8: # set of selected neighbour nodes of the root node
9: $SN \leftarrow \mathbf{0}$;
10: $UN \leftarrow \mathbf{0}$; # set of unselected nodes
11: $SUN \leftarrow \mathbf{0}$; # set of selected + unselected nodes
12: $V^v_{\gamma sort} \leftarrow sort(V^v_\gamma)$; # according to MCRR-LA
13: $V^s_{\gamma sort} \leftarrow sort(V^s_\gamma)$; #
14: $root\_node \leftarrow find\_root\_node(N^s, V^s_{\gamma sort})$;
15: $average\_degree \leftarrow compute\_average\_degree(G^s)$;
16: $search\_dist \leftarrow \left\lfloor \frac{|N^v|}{average\_degree} \right\rfloor$;
17: **while** $|SN| < |N^v|$ **do**
18: # CPU of selected nodes is $\geq \min_{n^v \in N^v} C^v(n^v)$
19: $SN \leftarrow BFS(G^s, root\_node, search\_dist)$;
20: **if** $|SN| < |N^v|$ **then**
21: $search\_dist = search\_dist + 1$;
22: **end if**
23: **end while**
24: $SN_{sort} \leftarrow sort(SN, V^s_{\gamma sort})$; # accor. to $V^s_{\gamma sort}$
25: $UN \leftarrow N^s \setminus SN$;
26: $UN_{sort} \leftarrow sort(UN, V^s_{\gamma sort})$; # accor. to $V^s_{\gamma sort}$
27: $SUN \leftarrow SN_{sort} + UN_{sort}$;
28: **for all** *nodes* n^v *ordered according* $V^v_{\gamma sort}$ **do**
29: **for all** *unused nodes* $n^s \in SUN$ **do**
30: **if** $C^v(n^v) \leq C^s(n^s)$ **then**
31: $counter = counter + 1$;
32: $M_{node}(n^v) = n^s$; # map n^v to n^s
33: **break**;
34: **end if**
35: **end for**
36: **end for**
37: # if all the virtual nodes are successfully mapped
38: **if** $counter = |N^v|$ **then**
39: $nodes\_mapping\_success \leftarrow true$;
40: update the CPU capacities C^s;
41: **else**
42: mark the VNR as rejected;
43: **end if**

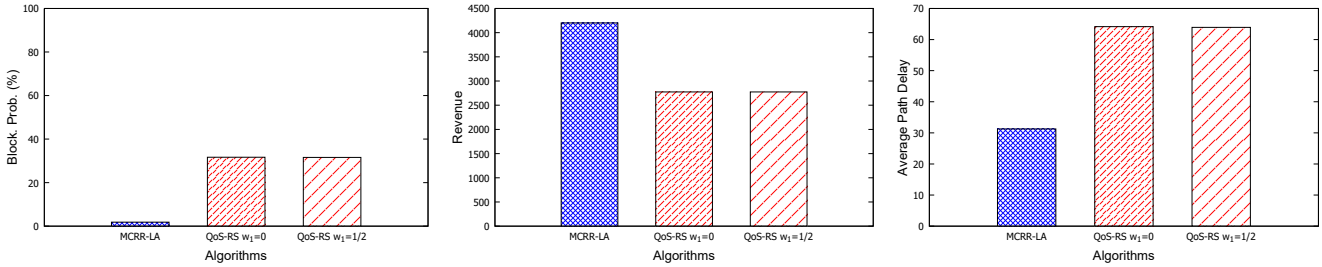
5. EXPERIMENTAL EVALUATION

We have evaluated our ranking algorithm through numerical evaluation. We have written a VNE simulator which im-



(a) Blocking Probability (b) Revenue (c) Average Path Delay

Figure 2: MCRR-LA vs QoS-RS: requested basic latency bounds · $\delta = 1.5$.



(a) Blocking Probability (b) Revenue (c) Average Path Delay

Figure 3: MCRR-LA vs QoS-RS: requested basic latency bounds · $\delta = 3$.

plements the proposed MCRR-LA algorithm and, for comparison, the algorithm proposed in [4][3], hereafter referred to as QoS-RS (QoS-Resource Selection) for brevity. We report the results based on a 32 node substrate network with 58 links (ANSNET [15], slightly modified as in [23]).

We assume that the VNR arrival process complies with a Poisson process with rate $\lambda = 5$. VNR lifetime is exponentially distributed with average lifetime T_{VNR} equal to 12 time units, corresponding to a medium load scenario. Every VNR is a random graph with a number of nodes uniformly distributed in $\{4, \ldots, 10\}$ with varying link connectivity rate in the different experiments. The VNR CPU, bandwidth requirements are randomly drawn node per node and link by link in the sets $\{4, \ldots, 8\}$ and $\{2, \ldots, 5\}$, respectively. The substrate network physical resources in terms of CPU and bandwidth are normalized and all equal to the nominal value of 100. The substrate network links latency are proportional to the geographical distance between nodes. The basic VNR latency requirements over the links are randomly generated in the interval between the average and the maximum latency value of the substrate network links times a multiplicative factor δ.

For the following results, every simulation experiment was 5000 time units long and was repeated 10 times. For the sake of clarity, we do not report the confidence interval which were in general not considerable.

In the experiments we compare MCRR-LA with QoS-RS using VNRs randomly generated with link connectivity rate of $\frac{ln(|N^v|)}{|N^v|-1}$ (to ensure that each node has $O(ln(|N^v|))$ incident edges). We tested the two algorithms under different ranges of latency VNR demands. Following [6], for MCRR-LA we adopt a value of $\gamma = 0.98$ (the discount factor of the Markov Reward Model). In the implementation of the QoS-RS algorithm, we do not consider the node location constraint and the links packet loss constraint which have no counterpart in our proposed solution (we plan to include such constraints in our future work). The K-Shortest-Paths algorithm used by QoS-RS is set with $K = 4$. In order to compute the quality of the physical paths, QoS-RS contemplates the use of weights (summing to 1), as in [21], to balance the importance of the bandwidth (w_1) and delay (w_2) attributes. Since our tests showed that the performance is not considerably affected by the different combination of the weights and since MCRR-LA aims to reduce the latency when it comes to compute the shortest paths, we plot the results of just two combinations: $w_1 = 0.5, w_2 = 0.5$ and $w_1 = 0.0, w_2 = 1.0$.

In Figures 2(a)–(c) we plot the VNR blocking probability, the average revenue and the average path delay of the algorithms, assuming the VNR latency bounds with a multiplicative factor $\delta = 1.5$ (that is, each VNR link latency requirement is randomly drawn in the interval $[1.5 \cdot average\_lat, 1.5 \cdot max\_lat]$, where $average\_lat$ and max_lat are the average and maximum substrate network link latency, respectively). From the figure, we can observe that MCRR-LA considerably outperforms QoS-RS in terms of lower blocking probability, higher revenue and lower average path delay. From the simulations, the lower blocking probability (also reflected in terms of higher revenue) is due to a more uniform use of the substrate network resources (both nodes and links) by MCRR-LA. Also MCRR-LA allows to reduce significantly the average path delay of the mapped VNRs compared to QoS-RS. Similar, albeit less pronounced differences can be observed with VNR latency bounds with a multiplicative factor $\delta = 3$ (corresponding to less stringent VNR links delay bounds) as shown in Figures 3(a)–(c).

Algorithm 2 *LinksMapping*

Input : VNR $G^v(N^v, E^v, C^v, B^v, L^v)$, Substrate net $G^s(N^s, E^s, C^s, B^s, L^s)$, mapping of nodes M_{node};
Output: Mapping of links M_{link};

$M_{link} \leftarrow \mathbf{0}$;
$links\_mapping\_success \leftarrow true$;
for all *virtual links* $e^v = (m^v n^v) \in E^v$ **do**
  $G^s_{clone} \leftarrow G^s$; # clone the substrate network
  # pre-cut links in G^s_{clone} without enough bandwidth
  for all *physical links* $e^s \in E^s_{clone}$ **do**
    if $B^s(e^s) < B^v(e^v)$ **then**
      cut link $e^s = (m^s n^s) \in E^s_{clone}$;
    end if
  end for
  compute a shortest path, $p^s(M_{node}(m^v), M_{node}(n^v))$, between the two nodes $M_{node}(m^v)$ and $M_{node}(n^v)$ in G^s_{clone}, in terms of latency;
  # if a shortest path, in terms of latency, was found
  # and it meets the requested latency constraint
  if $|p^s(M_{node}(m^v), M_{node}(n^v))| \neq 0$ **and** $p^s(M_{node}(m^v), M_{node}(n^v)).tot\_lat \leq L^v(e^v)$ **then**
    # mapping
    $M_{link}(m^v n^v) \leftarrow p^s(M_{node}(m^v), M_{node}(n^v))$;
    update the bandwidth B^s of involved links in G^s;
  else
    $links\_mapping\_success \leftarrow false$;
    mark the VNR as rejected;
    break;
  end if
end for
if $links\_mapping\_success = false$ **then**
  undo the CPU and bandwidth updates to G^s;
end if

The better performance of MCRR-LA can be ascribed to the use of latency aware metric in determining the node ranking which gives preference during the node mapping phase not only to the node with more resources (computational capacity and link bandwidth) but also with low latency. Indeed, despite the fact that QoS-RS uses a method to determine the quality of the paths during the link mapping, the nodes mapping phase is first carried out without considering latency. These results stress the importance of including link related QoS metric in the node mapping phase of two-stage mapping algorithms.

6. CONCLUSIONS

In this paper, we have presented a new approach to the Virtual Network Embedding problem based on Markov Reward Processes, the aim of which is to achieve a good tradeoff between resource utilization and QoS (e.g., latency). The main idea is to calculate the embedding potential of each node, that is aggregate resources in their respective neighborhood in terms of accumulated reward of a suitable random walk with the node as initial state. For both physical and virtual nodes, this value is penalized proportionally by the latency over the links. Based on this new metric, we proposed a two-stage VNE algorithm which ranks virtual nodes according to the previously computed latency aware metric, and physical nodes according to the previously computed latency aware metric combined with a Breadth-First search on the highly ranked node. Subsequently, the algorithm carries out the node mapping accordingly and finally performs the link mapping utilizing the shortest path in terms of latency. Experimental outcomes show that our algorithm is able to reduce the average path delay of the embedded VNRs compared to alternative approaches. As future work, we plan to improve the algorithm acting both on nodes and links mapping stages, to execute more experiments in realistic setting, and to extend MCRR-LA to account for additional VNR QoS requirements, including network resiliency.

7. REFERENCES

[1] D. G. Andersen. Theoretical approaches to node assignment. Unpublished Manuscript, Dec. 2002.

[2] M. T. Beck and C. Linnhoff-Popien. On delay-aware embedding of virtual networks. In *The sixth international conference on advances in future internet, AFIN*, 2014.

[3] S. Behrouznia. A qos-based resource selection approach for virtual networks. Master's thesis, Concordia University, April 2015.

[4] S. Behrouznia, R. Dssouli, and M. El Barachi. A qos-based resource selection approach for virtual networks. In *International Conference on Computer and Information Science and Technology, 2015. CIST'15.*, May 2015.

[5] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

[6] F. Bianchi and F. Lo Presti. A markov reward model based greedy heuristic for the virtual network embedding problem. In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2016 IEEE 24th International Symposium on*, 2016.

[7] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, Apr. 1998.

[8] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang. Virtual network embedding through topology-aware node ranking. *SIGCOMM Comput. Commun. Rev.*, 41(2):38–47, Apr. 2011.

[9] N. Chowdhury, M. Rahman, and R. Boutaba. Virtual network embedding with coordinated node and link mapping. In *INFOCOM 2009, IEEE*, pages 783–791, April 2009.

[10] N. Feamster, L. Gao, and J. Rexford. How to Lease the Internet in Your Spare Time. *SIGCOMM Comput. Commun. Rev.*, 37(1):61–64, Jan. 2007.

[11] A. Fischer, J. Botero, M. Till Beck, H. de Meer, and X. Hesselbach. Virtual network embedding: A survey. *Communications Surveys Tutorials, IEEE*, 15(4):1888–1906, Fourth 2013.

[12] R. G. Gallager. *Stochastic processes: theory for applications*. Cambridge University Press, Cambridge, 2013.

[13] L. Gong, Y. Wen, Z. Zhu, and T. Lee. Toward profit-seeking virtual network embedding algorithm via global resource capacity. In *INFOCOM, 2014 Proceedings IEEE*, pages 1–9, April 2014.

[14] K. Ivaturi and T. Wolf. Mapping of delay-sensitive virtual networks. In *Computing, Networking and Communications (ICNC), 2014 International Conference on*, pages 341–347, Feb 2014.

[15] Z. Li and J. J. Garcia-Luna-Aceves. Finding multi-constrained feasible paths by using depth-first search. *Wirel. Netw.*, 13(3):323–334, June 2007.

[16] 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*, VISA '09, pages 81–88, New York, NY, USA, 2009. ACM.

[17] J. Lu and J. Turner. Efficient Mapping of Virtual Networks onto a Shared Substrate. Technical Report 2006-35, Washington University in St. Louis, 2006.

[18] L. Nonde, T. E. H. El-Gorashi, and J. M. H. Elmirghani. Energy efficient virtual network embedding for cloud networks. *Journal of Lightwave Technology*, 33(9):1828–1849, May 2015.

[19] M. R. Rahman and R. Boutaba. Svne: Survivable virtual network embedding algorithms for network virtualization. *IEEE Transactions on Network and Service Management*, 10(2):105–118, June 2013.

[20] A. Razzaq and M. Rathore. An approach towards resource efficient virtual network embedding. In *Evolving Internet (INTERNET), 2010 Second International Conference on*, pages 68–73, Sept 2010.

[21] J. Shamsi and M. Brockmeyer. Qosmap: Qos aware mapping of virtual networks for resiliency and efficiency. In *2007 IEEE Globecom Workshops*, pages 1–6, Nov 2007.

[22] M. Yu, Y. Yi, J. Rexford, and M. Chiang. Rethinking virtual network embedding: Substrate support for path splitting and migration. *SIGCOMM Comput. Commun. Rev.*, 38(2):17–29, Mar. 2008.

[23] S. Zhang, Z. Qian, J. Wu, and S. Lu. An opportunistic resource sharing and topology-aware mapping framework for virtual networks. In *INFOCOM, 2012 Proceedings IEEE*, pages 2408–2416, March 2012.

[24] Y. Zhu and M. Ammar. Algorithms for assigning substrate network resources to virtual network components. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, April 2006.