IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 8, 2018, accepted November 23, 2018, date of publication December 4, 2018,
date of current version December 31, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2885033

Location Aware and Node Ranking Value-Assisted Embedding Algorithm for One-Stage Embedding in Multiple Distributed Virtual Network Embedding

HAOTONG CAO[1,2,3], (Student Member, IEEE), YONGAN GUO[1,2,3], (Member, IEEE), YUE HU[4], SHENGCHEN WU[5], (Student Member, IEEE), HONGBO ZHU[1,2,3], AND LONGXIANG YANG[1,2,3]

[1]Key Laboratory of Broadband Wireless Communication and Sensor Network Technique, Ministry of Education, Nanjing University of Posts and Telecommunications, Nanjing 210003, China
[2]Jiangsu Key Laboratory of Wireless Communications, Nanjing University of Posts and Telecommunications, Nanjing 210003, China
[3]College of Telecommunications and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China
[4]Internet of Things Research Center, China Mobile Communications Group Jiangsu Co., Ltd., Nanjing 210003, China
[5]College of Overseas Education, Nanjing University of Posts and Telecommunications, Nanjing 210023, China

Corresponding author: Longxiang Yang (yanglx@njupt.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFC1314903, in part by the National Natural Science Foundation of China under Grant 61372124 and Grant 61427801, and in part by the Postgraduate Research and Practice Innovation Program of Jiangsu Province under Grant KYCX17_0784.

ABSTRACT Network virtualization (NV) is one crucial attribute for the next-generation network. Virtual network embedding (VNE) is known to be the resource allocation problem in NV content. Since 2008, researchers have proposed multiple embedding algorithms to embed VNs onto the underlying physical networks. Considering the NP-hard nature of VNE, most of the prior algorithms focus on embedding each VN in two separated embedding stages (separated node and link embeddings). Some embedding algorithms embed each VN in one embedding stage by adopting a mixed integer linear programming approach or game theory or sub-graph isomorphism, trapped in high computation time. These algorithms embed the VN in one centralized substrate network (SN), while, in real networking environment, each VN must be embedded among multiple SNs. Each substrate node is geographically distributed in a different location. In order to address these issues, we propose a Location-Aware and Node Ranking Value Assisted embedding algorithm (labeled as LANRVA-VNE). The LANRVA-VNE conducts each VN embedding in one embedding stage. In addition, the LANRVA-VNE is able to embed each requested VN service among multiple distributed SNs in polynomial time, aiming at promoting future dynamic VN service implementation. Main evaluation results reveal that the LANRVA-VNE significantly improves VN acceptance ratio by at least 10% over the typical two-stage algorithms (e.g., VNE-NTANRC-S, RW-SP, and ViNE-SP).

INDEX TERMS Next generation network, virtual network embedding, one embedding stage, multiple distributed substrate networks, polynomial time, future dynamic VN service implementation.

I. INTRODUCTION

Current communication networks employ the 'one-size-fits-all' approach to serve all end users, regardless of the concrete requirements of vertical services [1]. However, this design approach can't offer different services and fulfill explosive data traffic requests [2]. Thus, the next generation network must explore new techniques in order to fulfill various vertical service requirements. Till now, researchers have reached the consensus that network virtualization (NV) is one crucial technique of the next generation network [3], particularly to the upcoming 5G virtualized mobile network. Through virtualization, multiple customized and different virtual network (VN) requests are allowed to be scheduled and implemented among multiple distributed substrate networks (SNs) for providing various vertical services (e.g. a broadcat service with high communication bandwidth) seamlessly [3] and efficiently.

In NV content, conventional Internet service providers (ISPs) will be decoupled into two isolated roles: the infrastructure providers (InP) (e.g. China Tower [4]) and service

providers (SPs) (e.g. Chine Mobile, Wangyi, Tencent). This decoupling scheme intends to make the most use of underlying resources and provide different network services [3]. Isolated SPs are responsible for implementing requested VNs so as to fulfill various resource/functional demands of end users. End users will pay for their requested VN services. Thus, SPs earn benefits from end users. These benefits are accepted as the embedding revenues of SPs. In this business model [3], SPs rent various underlying substrate resources (e.g. CPU, node storage, communication bandwidth) from the InP. Consequently, SPs must pay to the InP, on the basis of their contract agreements. The InP will expose part or all information of his resource(s) to rented SPs, such as his network topology, resource amount. In this paper, we study the case of SPs knowing all the rented resource information of the InP. It is owning to the fact that the InP will update his resource information after the contract expired [7]. Thus, it is feasible for the InP to expose his resource information [5] to his rented SPs. On the whole, it is vital for SPs to embed more requested VNs onto their rented underlying SNs in order to earn more embedding revenues and make the most use of rented underlying resources. With more VNs requested and accommodated, SPs need to rent extra underlying resources from the InP. Consequently, SPs will pay more revenues to the InP. Therefore, the InP will earn more embedding revenues.

Based on these backgrounds, we can discover the fact that it is rather vital for SPs and the InP to be equipped with the embedding algorithms. These embedding algorithms contribute to fulfilling requested VN services optimally, according to end users' concrete VN service goals. While in the literature, the problem of embedding VNs onto underlying SNs is so-called virtual network embedding (VNE). Consequently, these embedding algorithms are called as VN embedding algorithms [5], [7].

Since 2008, researchers have proposed multiple VN embedding algorithms to deal with VNE problem. Multiple surveys [5]–[7] have been conducted to detail VN embedding algorithms in recent years. Considering the NP-hard nature of VNE [6], researchers intend to propose feasible embedding algorithms. In the literature, feasible embedding algorithms include greedy node embedding strategy and shortest substrate path (SP) embedding [7], ELECTRE method based embedding [8], mixed integer linear programming [9], column generation aided linear programming [10], topology driven and network resources combined embedding [11], [12], candidate sets assisted integer linear programming [13] and other heuristics (e.g. [14], [15]). However, most of prior algorithms (e.g. [8], [10]–[12]) focus on embedding each VN in **t**wo **s**eparate stages (separated node embedding and link embedding). Some embedding algorithms (e.g. [6], [9], [14], [15]) embed each VN in one embedding stage by using mixed integer linear programming (MILP) approach [9] or game theory [14] or subgraph isomorphism [15]. This kind of embedding algorithms will be trapped in high computation time. In addition, these algorithms embed each VN in **O**ne **C**entralized SN. Thus not being promoted to future virtual network implementation. Though *CAN-A* [13] proves to conduct VN embedding in polynomial time, it conducts the VN embedding in **O**ne **C**entralized SN and hence, not being promoted to real network implementation. In real networking environment, each VN must be embedded among multiple distributed SNs. Each substrate node of SNs is geographically distributed in a different location. This kind of VN embedding problem is called **M**ultiple **D**istributed VNE [5]–[7]. Though some embedding algorithms [16]–[18] have been proposed in recent years, they are two-stage embedding algorithms [17] and time-consuming [18], leading to local underlying resources overloaded and low VN acceptance. With respect to [16], the adopted MILP approach has a large computation complexity. Though ensuring exact VN embedding solution, it cannot be promoted to real network environment for time sensitive services [6] (e.g. online gaming service).

On the account of above backgrounds, we propose a **L**ocation **A**ware and **N**ode **R**anking **V**alue **A**ssisted embedding algorithm (labeled as *LANRVA-VNE*) to deal with the **M**ultiple **D**istributed VNE problem. Our primary embedding goal is to accommodate as many VNs as possible. Thus maximizing the embedding revenues of the InP and SPs. We firstly construct the formal **M**ultiple **D**istributed VNE problem model. Then, we detail our proposed *LANRVA-VNE* algorithm. Our *LANRVA-VNE* is able to embed the requested VN in **o**ne **e**mbedding stage, among **m**ultiple **d**istributed SNs. Before conducting the embedding, our *LANRVA-VNE* will construct the candidate node sets for all virtual nodes, according to VN nodes' **L**ocation requirements/awareness. With respect to any virtual node M, our *LANRVA-VNE* will rank all its candidate substrate nodes in an iterative approach, using our calculated **N**ode **R**anking **V**alues. In running VN embedding, the greedy node and minimum intermediate nodes preferred shortest-path (SP) embedding scheme is adopted. Our *LANRVA-VNE* considers CPU, node storage, desired location, communication bandwidth, maximum virtual link allowed intermediate nodes and link propagation delay as VN node-link constraints. Most of previous VNE algorithms just execute CPU and communication bandwidth (e.g. [5]–[7]) constraints while embedding requested VNs. To further highlight the *LANRVA-VNE* efficiency, we conduct the evaluation work. Main evaluation results are illustrated, revealing that our *LANRVA-VNE* significantly improves VN acceptance ratio by at leat 10% over the typical two-stage embedding algorithms (e.g. *VNE-NTANRC-S* [12], *RW-SP* [11], *ViNE-SP* [8]). Other performance metrics (e.g. revenue to cost ratio) are plotted to strengthen our *LANRVA-VNE* efficiency, too.

The rest of this paper is organized as follows. Formal **M**ultiple **D**istributed VNE problem model is described in Section II. We detail our *LANRVA-VNE* algorithm in Section III. The evaluation work is conducted in Section IV. In the end, we conclude this paper.

II. FORMAL MULTIPLE DISTRIBUTED VNE PROBLEM MODEL

A. FORMAL MULTIPLE DISTRIBUTED VNE PROBLEM MODEL

In VNE research area, most researchers adopt the graph theory [7] to model the SN and VN. While in **M**ultiple **D**istributed VNE, we still use the graph theory to model the distributed SN and VN. The whole underlying substrate network ($G^s_{whole} = (N^s_{whole}, L^s_{whole})$), owned by the isolated InP (e.g. China Tower), is composed of $|N|$ geographically distributed SNs. The ith SN ($i = 1, 2, 3 \ldots |N|$) is represented by the weighted graph $G^s_{ith} = (N^s_{ith}, L^s_{ith}, A^s_{ith})$. N^s_{ith} represents all underlying physical nodes of the ith SN. L^s_{ith} refers to all underlying physical links of the ith SN. A^s_{ith} represents all substrate element attributes of the ith SN, listed in Table 1. In Fig. 1, we also plot the whole substrate network that consists of 2 geographically distributed SNs (SN 1 and SN 2). Substrate links, connecting distributed SNs, are set to have enough communication bandwidths for long distance communication, reserved by the InP. Available communication bandwidths in each isolated SN refer to numbers over links. Available CPUs and node storages refer to numbers in rectangles. For clarity, remaining network elements, such as physical node location, are not plotted.

TABLE 1. *i*th SN substrate element attributes.

Substrate Node, CPU Attribute	C^s_m
Substrate Node, Storage Attribute	S^s_m
Substrate Node, Required Location Attribute	$loc(m)$, x, y coordinates
Substrate Bandwidth Attribute	B^s_{mn}
Link mn Propagation Delay Attribute	PD^s_{mn}
Substrate Paths Set of G^s_{whole}	P^s_{whole}
Substrate Paths Set of G^s_{ith}	P^s_{ith}
All Direct Paths from Node m to n in P^s_{whole}	P^s_{mn}
One Selected Path from P^s_{mn} Set	P_{mn}

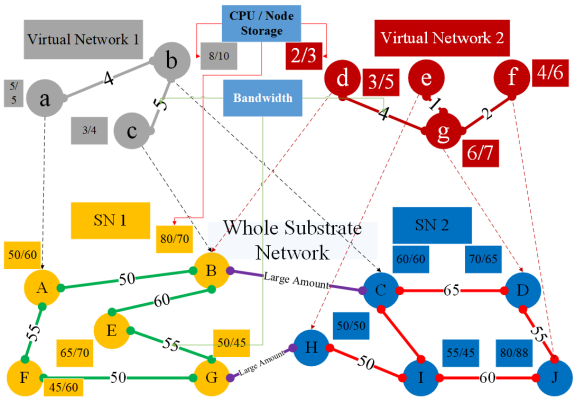


FIGURE 1. An Example of Multiple Distributed VNE and Two VNs Embedding.

Similarly, we model each VN request in the form of $G^V = (N^V, L^V, A^V)$. N^V represents all virtual nodes of the VN. L^V refers to all virtual links in the VN. A^V refers to all virtual element attributes, listed in Table 2. We draw two different VN requests (VN 1 and VN 2) in the Fig. 1. Required communication bandwidths refer to numbers over links. CPUs and node storages refer to numbers in rectangles. For clarity, desired location, maximum allowed intermediate substrate nodes and maximum allowed virtual link propagation delay are omitted from the Fig. 1.

TABLE 2. VN network element attributes.

Virtual Node, Required CPU Attribute	C^V_M
Virtual Node, Storage Demand Attribute	S^V_M
Virtual Node, Required Desired Location	$Loc(M)$
Virtual Link, Required Bandwidth Attribute	B^V_{MN}
Virtual Link, Allowed Propagation Delay	PD^V_{MN}
Maximum Allowed Distance Deviation of M	$LR(M)$
Maximum Allowed Intermediate Nodes of MN	$IN(MN)$

With regards to introducing the whole substrate network and VN request models, we decide to describe the VN embedding onto the whole underlying network. With respect to the embedding of VN, two embedding procedures are usually involved: first node embedding procedure and subsequent link embedding procedure. Both embedding procedures are shown below.

1) NODE EMBEDDING PROCEDURE

For completing node embedding, all VN virtual nodes are embedded by using the unqiue node embedding function $\boldsymbol{F}_N(): N^V \rightarrow N^S_{whole}$.

$$\boldsymbol{F}_N(\boldsymbol{M}) \in N^s_{whole}$$
$$\boldsymbol{F}_N(\boldsymbol{M}) = \boldsymbol{F}_N(\boldsymbol{N}), \ \textit{if and only if} \ \boldsymbol{M} = \boldsymbol{N}$$

subject to

$$\textit{All Node Constraints are Satisfied} \tag{1}$$

where all virtual node constraints (CPU, node storage and node location considered in this paper) of the VN must be fulfilled in Expression 1. Otherwise, the VN will be rejected. For example, in Fig. 1, virtual nodes a, b and c of VN 1 are embedded onto substrate nodes A, C and B, by fulfilling all node resource and functional constraints. With respect to the VN 2, all virtual nodes are successfully embedded: $d \rightarrow B$, $e \rightarrow H$, $f \rightarrow J$ and $g \rightarrow D$. At the same time, all node constraints of VN 2 are fulfilled.

2) LINK EMBEDDING PROCEDURE

In subsequent link procedure, we perform link embedding by running the unique link-embedding function $F_L(): L^V \rightarrow P^s_{whole}$, for all VN virtual links. During the link embedding procedure, path splitting and link interference are usually not permitted, aiming at not affecting the VN quality of experience (QoE) [5], [7] performance.

$$\boldsymbol{F}_L(\boldsymbol{MN}) \subseteq P^s_{whole}(\boldsymbol{F}_N(\boldsymbol{M})\boldsymbol{F}_N(\boldsymbol{N}))$$

subject to

$$All\ Link\ Constraints\ are\ Satisfied \tag{2}$$

where all VN link constraints (communication bandwidth demand, maximum virtual link allowed intermediate nodes and required link propagation delay in this paper) of any VN link must be satisfied in Expression 2. Otherwise, the VN will be rejected. For instance, virtual link *ab* of VN 1 is embedded onto the substrate path *ABC* (Fig. 1), by fulfilling all link resource and functional constraints. For virtual link *bc*, it is embedded onto the substrate path *CB*. With respect to VN 2, all its virtual links are embedded successfully: $gd \rightarrow DCB$, $ge \rightarrow DCIH$ and $gf \rightarrow DJ$. All link (resource and functional) constraints are fulfilled, too.

With the above two embedding procedures completed successfully, the VN embedding is done. In VNE research, most embedding algorithms [5] conduct the VN embedding in two separate procedures. Node embedding procedure is conducted firstly. Then follows the subsequent link embedding procedure. No or part coordination of two procedures exists. Thus, the embedding quality cannot be ensured in most cases, using this embedding scheme [6]. Thus calling for proposing one-stage embedding algorithm, combining node and link embeddings into one stage and improving the VN embedding quality. In addition, the VN embedding execution time must be minimized. We must get rid of the occasion where the embedding time is trapped in high computation time [6], [7]. We intend to promote the VN embedding to future dynamic and real-time VN service implementation.

B. MAIN PERFORMANCE EVALUATION METRICS FOR MULTIPLE DISTRIBUTED VNE

To evaluate the **M**ultiple **D**istributed VNE algorithms, we formulate main performance metrics in this subsection. First and foremost, it is the embedding revenue by accepting a VN, earned by the implemented SP.

$$\boldsymbol{Rev}\left(G_T^V\right) = \begin{cases} \boldsymbol{Rev}\left(G^V\right) \cdot T, & if\ \mathbf{G}^V\ is\ embedded \\ 0, & else \end{cases} \tag{3}$$

where T refers to embedded VN lifespan. Similar to [8]–[18], we formulate the per-unit revenue of the $Rev(G^V)$ (Expression 4). It is not a nonlinear function [5]–[7] and is equal to total virtual CPUs, node storages and virtual communication bandwidths. Weights method (α, β and γ) is adopted in order to balance various VN node and link constraints.

$$\boldsymbol{Rev}\left(G^V\right) = \alpha \cdot \sum_{M \in N^V} \mathbf{C}_M^V + \beta \cdot \sum_{M \in N^V} \mathbf{S}_M^V + \gamma \cdot \sum_{MN \in L^V} \mathbf{B}_{MN}^V \tag{4}$$

Though Expressions 3 and 4 indicate how much embedding revenues can be earned, it is meaningless not to measure the consumed CPU, storage and bandwidth resources. We formulate the Expression 5 so as to represent the VN per-unit cost. Weight factors intend to balance different resources, same as Expression 4. $\boldsymbol{H}_{MN}^{mn}$ records the number of hops in the underlying path ***mn*** that accommodates the virtual link ***MN***. Note that the term 'hops' has the same meaning with the term 'intermediate nodes' in this paper.

$$\boldsymbol{Cos}\left(G^V\right) = \alpha \cdot \sum_{M \in N^V} \mathbf{C}_M^N + \beta \cdot \sum_{M \in N^V} \mathbf{S}_M^N + \gamma \cdot \sum_{MN \in L^V} \sum_{mn \in P_{mn}^s} \boldsymbol{H}_{MN}^{mn} \cdot \mathbf{B}_{MN}^V \tag{5}$$

$$\boldsymbol{Cos}\left(G_T^V\right) = \begin{cases} \boldsymbol{Cos}\left(G^V\right) \cdot T, & if\ \mathbf{G}^V\ is\ embedded \\ 0, & else \end{cases} \tag{6}$$

Similar to Expression 3, we formulate Expression 6. This expression enables us to calculate the amount of consumed substrate resources of the accepted VN, during the VN lifetime. Pertaining to the net profit by embedding the VN, it is the result of VN embedding revenue minus VN embedding cost. With respect to other revenues and costs, such as energy cost, we do not add them to calculate the net profit in this paper. Therefore, VN embedding revenue and embedding cost are selected as the main evaluation metrics so as to evaluate our *LANRVA-VNE*.

The VN acceptance ratio, which will be illustrated in Section IV, is a vital performance metric for evaluating various VNE algorithms' embedding abilities in NV research. With respect to node storage utilization, link communication bandwidth utilization and VN embedding execution time, they are illustrated in Section IV, too. However, we do not introduce and detail these performance metrics in this paper. This is owing to the fact that we concentrate on detailing our *LANRVA-VNE* and highlighting *LANRVA-VNE* performance efficiency. We have made a detailed description work of VNE performance metrics in our recent published surveys [6] and [7]. Any interested reader can refer to.

III. DETAILS OF LANRVA-VNE

We detail the *LANRVA-VNE* in this section. For better understanding the algorithm *LANRVA-VNE*'s details, we will use our *LANRVA-VNE* algorithm to conduct a given VN embedding.

A. CONSTRUCT CANDIDATE SETS FOR ALL VN VIRTUAL NODES

While receiving a VN request, $G^V = (N^V, L^V, A^V)$, from an end user, we will construct the candidate substrate node sets for all virtual nodes in the given G^V in the first place.

With respect to any selected virtual node M, it is provided with a required node location, $Loc(M) = (X_M, Y_M)$, and maximum allowed distance deviation $LR(M)$. For any substrate node m in the whole substrate network N_{whole}^S, it has its location $Loc(m) = (X_m, Y_m)$. Thus, we formulate Expression 7. Dis_{Mm}^s records the ***Euclidean Distance*** between virtual node M and substrate node m. If Expression 7 is satisfied, we will further compare m with M, in terms of CPU and node storage demands. If m has reserved enough CPU and node storage resources for accommodationg the VN node M, we will select node m to be one candidate substrate node for

VN node M. With respect to remaining substrate nodes in the whole substrate network N^S_{whole}, we will select all suitable candidate substrate nodes and form up the candidate node set for VN node M.

To remaining $(|N^V|-1)$ virtual nodes (e.g. virtual node N) in G^V, we repeat the above procedure for $(|N^V|-1)$ times. In total, the number of candidate sets for all VN nodes is equal to $|N^V|$.

$$\boldsymbol{Dis}^s_{Mm} = \sqrt{(X_M - X_m)^2 + (Y_M - Y_m)^2} \leq LR(M) \quad (7)$$

However, we must take notice of an exception that different virtual nodes belonging to the same VN (VN node M, VN node N and VN node A) may share the same underlying node (e.g. substrate node m). To deal with this exception, we will assign the substrate node to the virtual node that has the fewest $Sum(\ )$ value. See our Expression 8 for $Sum(\ )$ formulation.

$$\mathbf{Sum(M)} = \mathbf{C^V_M} + \mathbf{S^V_M} \quad (8)$$

B. EFFICIENT NODE RANKING APPROACH BY ITERATION

With constructing candidate substrate node sets for all virtual nodes, it follows the procedure of most suitable substrate node selection. With respect to the virtual node M, it is of great importance to select the most suitable underlying physical node m to embed the VN node M. In previous VNE research, researchers have found that node ranking approach contributes to improving the following VN embedding quality and accommodating more VNs. To the virtual node M, the selected and suitable candidate node m has the highest node ranking value among its all candidate nodes, enabling to coordinate node and link embeddings. Thus increasing the eventual embedding revenue of SPs and InP eventually.

Based on this discovery and our latest research [12], we propose the efficient node ranking approach, enabling to accurately **N**ode **R**anking **V**alue in an iterative approach. Our node ranking approach is proposed for solving **M**ultiple **D**istributed VNE Problem. Concrete procedures are shown below.

1) SELECT AND QUANTIFY TOPOLOGY ATTRIBUTES AND NETWORK RESOURCES

In our latest research [12], we have proved that each topology attribute has its effect on embedding quality. However, it is unnecessary to select too many topology attributes. Therefore, we just select three basic attributes (node degree, node strength and node farness) and two derived attributes in our node ranking approach. Derived attributes are node centrality and link interference. Due to limited pages, detailed topology definitions are not provided in our paper. Any interested reader can refer to [19] and our latest publication [12]. We focus on introducing the procedures of achieving **N**ode **R**anking **V**alue for selected candidate substrate node.

For any node m in the network, we define $De(m)$, $Str(m)$ and $Far(m)$ to denote its node degree, node strength and node farness attributes. With respect to $De(m)$, it measures the amount of direct links connecting physical node m. $Str(m)$ records all direct link communication bandwidths of node m. With respect to $Far(m)$, we formulate Expression 9 in order to count all shortest ***Euclidean Distance*** (Expression 7) from node m to all remaining nodes.

$$\boldsymbol{Far}\ (m) = \sum_{m,n\in N^s, m\neq n} \mathbf{Dis}^s_{mn} \quad (9)$$

To two derived attributes, node centrality (Expression 10) records the number of times that m serves as a bridge along the shortest path between any two different nodes (e.g. a and b). With respect to the link interference $LI(mn)$ attribute, we formulate Expression 11. This attribute is related to the contribution to measuring the network connectivity. Take note that we select $LI(mn)$ serving as the only link attribute for calculating node values. While in our published [12], the link attribute is not selected for calculating **N**ode **R**anking **V**alues.

$$\boldsymbol{NoCe}\ (m) = \sum_{m,a,b\in N^s, m\neq a\neq b} \frac{Num(a,b)(m)}{Num(a,b)} \quad (10)$$

$$\boldsymbol{LI}\ (mn) = \frac{Far(m)}{De(m)} + \frac{Far(n)}{De(n)} \quad (11)$$

With respect to network resources considered, CPU C^s_m, node storage S^s_m and communication bandwidth B^s_{mn} are quantified in our node ranking approach.

2) CALUCLATE NODE RANKING VALUES

We firstly define a novel resource metric, called as Resource Block (RB). In VNE research, the RB metric can be seen as the extension of previous local resource product value [11] which served as the only ranking metric to rank nodes. We further formulate the RB metric of node m in Expression 12, along with its normalized version (Expression 13).

$$\boldsymbol{RB}\ (m) = C^s_m \cdot S^s_m \cdot \boldsymbol{Str}\ (m) \cdot \sum_{m,n\in N^s, m\neq n} LI\ (mn) \quad (12)$$

$$\boldsymbol{RB}\ (m)\ \% = \frac{\boldsymbol{RB}\ (m)}{\sqrt{\sum_{m\in N^s} RB(m)^2}} \quad (13)$$

Stimulated from the *Coulomb's law* in electromagnetism area, we derive the method of quantifying the relationship between two isolated discrete nodes (nodes m, n) (Expression 14). We need to take note that quantified nodes have to be in the same network. Otherwise, no logical and quantitative relationships between node m and node n exist.

$$\boldsymbol{ITA}\ (m,n) = \alpha \cdot \frac{S^s_m \cdot S^s_n}{{Dis^s_{mn}}^2} \quad (14)$$

Another notice is pointed out that at leat one underlying physical path must exist in order to connect m and n. Otherwise, the value of Expression 14 is 0. As Expression 14 just measures the m's distance relationship with one selected node n. Expression 15 is formulated, intending to sum up m

interactions with other different nodes in the whole network. Normalized version is formulated in Expression 16, too.

$$\boldsymbol{ITA}\ (m) = \boldsymbol{NoCe}\ (m) \cdot \sum_{m,n \in N^s, m \neq n} \boldsymbol{ITA}\ (m,n) \quad (15)$$

$$\boldsymbol{ITA}\ (m)\ \% = \frac{\boldsymbol{ITA}\ (m)}{\sqrt{\sum_{m \in N^s} ITA(m)^2}} \quad (16)$$

We calculate all normalized percentage values, using Expression 16. In addition, we use these percentages to make up an initial node value vector, labeled as T_0 in this paper. The vector T_0 has a dimension of $|N| \cdot 1$.

$$T_0 = (\boldsymbol{ITA}\ (1)\ \%, \boldsymbol{ITA}\ (2)\ \%, \ldots, \boldsymbol{ITA}\ (|N|)\ \%)^T$$

Then, we decide to calculate an eventual node value of m NRV_m^s in a recursive manner, stimulated from well-known *Markov* model approach [20] and previous published [11]. In Expression 17, $m = n + 1$.

$$r_m = (1-d) \cdot RB(m)\% + d \cdot \sum_{m \neq n, n \in N^s} ITA_{m,n} \cdot r_n \quad (17)$$

where d represents the so-called damping factor for *Markov* random model, within the range of (0,1). N^s refers to all loop-free paths directly linking to m. In order to represent all nodes'values in the form of final vector R, we further formulate all node ranking values' traffic form in the following expression.

$$R = (1-d) \cdot RB\% + d \cdot M \cdot R \quad (18)$$

where R is represented by $\left(\boldsymbol{r}_{(1)}, \boldsymbol{r}_{(2)}, \ldots, \boldsymbol{r}_{(|N|)}\right)^T$, after multiple recursive procedures. M in Expression 18 refers to the transition matrix, with a dimension of $|N| \cdot |N|$. Within the transition matrix M, each element is formulated in Expressions 15 and 16. This matrix has the function of quantifying distance relationship. For instance, if no underlying path connects m to n, the value of corresponding element is 0. We can easily get to calculate the unique solution of Expression 18 [21]. The unique node value solution is shown in Expression 19 below.

$$R = (1-d) \cdot (1 - d \cdot M)^{-1} \cdot RB\% \quad (19)$$

Concrete procedures of getting Expression 19 are not involved in this paper. We have made the detailed proof in [12]. Specially, the proof of the uniqueness of Expression 19 is involved, even if adding the link interference attribute and resource block (RB). Derived from [20], we can get to know that the time complexity of directly calculating out Expression 19 is at least $O(|N|^3)$. Time complexity grows exponentially with network scale expanding, no matter substrate or virtual network. Therefore, it is impractical to directly calculate Expression 19, considering promoting VN implementation to dynamic network environment.

Instead, we decide to use an iterative and conventional approach so as to calculate the near-optimal solution in limited time. In [19], iterative-based calculation is proved effective and efficient. Through k iterations, Expression 19 will converge to a stable solution [19]. Thus being seen as the near-optimal solution. Time complexity of the iteration-based calculation is decreased to $O((|N_{substrate\ or\ virtual}|^2) \cdot log(1/\delta))$. δ is a defined small positive number. It aims at limiting the number of iterations. Pseudo code is presented in **Algorithm 1**.

Algorithm 1 Iterative-Based Node Ranking Calculation

Require: Given Network $G = (N, L)$, a limited positive value δ

Ensure: Node Ranking Value Vector $\mathbf{R}$

```
Get matrix M, the initial vector R_0(or T_0)
Define an iteration number k with initial vaue 0.
Define a variable w, up to ∞.
while w ≥ δ do
    R_{k+1} = (1 − d) · RB% + d · M · R_k;
    w = ‖R_{k+1} − R_k‖;
    k = k + 1;
end while
R = R_{k+1}
```

Following above procedures, we can compute each node ranking value (NRV_m^s) in the end. To improve the algorithm efficiency, we conduct an extra action ahead. With respect to the VN node M, we just compute all node ranking values of its candidate nodes ($NRV_{m,M}^s$). Then, we sort all candidate SN nodes of M in an increasing manner. We intend to select the candidate SN node m with the highest $NRV_{m,M}^s$ value. As mentioned in Section III-A, the selected candidate SN node m has enough node resources for accepting the virtual node M.

C. SIMULTANEOUS NODE AND LINK EMBEDDING FOR EMBEDDING THE REQUESTED VN

With completing candidate set construction and iterative-based node ranking value calculation, it is the simultaneous node and link embedding procedure, conducting the VN embedding in one embedding stage. The simultaneous node and link embedding of *LANRVA-VNE* involves the following two sub-procedures:

1) INTER-DOMAIN NODE EMBEDDING

Considering different distributed SNs, this sub-procedure aims at selecting proper SN for each virtual node (e.g. M) of VN in order to improve our *LANRVA-VNE* efficiency. As the Expression 7 indicates node M desired location and allowed derivation, we select the proper SN, having suitable candidate substrate nodes (e.g. substrate node m). The SN node m must have highest node ranking value among all candidate physical nodes of M. To other remaining nodes in the VN, this section procedure is conducted $|N^V - 1|$ times. As we have assumed that links connecting isolated SNs have infinite communication bandwidths, we do not need to worry about Inter-Domain resource shortage. In addition, this sub-procedure can be conducted along with candidate set

construction procedure and node ranking value calculation procedure. Therefore, the time complexity is limited within $O((|N_{candidate\ SN}|^2) \cdot log(1/\delta))$.

2) INTRA-DOMAIN NODE AND LINK EMBEDDING

With completing the Inter-Domain node embedding, the subsequent Intra-Domain node and link embedding follows. We firstly backup resource information of whole substrate network. Then we rank all candidate physical nodes of the VN node M in a descending order, on the basis of the calculated node ranking values in Expression 19 (Section III-B). Usually, VNE researchers embed the virtual node M onto the physical node m with highest ranking value. The selected substrate node m must reserve available resources so as to meet all virtual node constraints. If any virtual node demand cannot be fulfilled by all candidate physical nodes, researchers will reject the VN. If all virtual nodes are accepted, researchers will update the SN physical information. We use our NAL (node and link) approach to conduct the Intra-Domain node and link embedding. Though our NAL runs in a similar greedy node embedding strategy, some differences exist. In our NAL, we firstly complete two virtual nodes', M and N, greedy embeddings, having first two highest virtual node ranking values. Two virtual nodes, M and N, are embedded onto two candidate substrate nodes, m and n. Node m and node n have the highest node ranking values in their own candidate sets. If there exists a virtual link connecting both virtual nodes, the virtual link will be simultaneously embedded, using minimum intermediate substrate nodes preferred SP method. For embedding the virtual link, we must select the suitable physical path, having enough communication bandwidth, allowed number of intermediate nodes and allowed link propagation delay. In this paper, the link propagation delay of the substrate link, connecting any two isolated SNs, is set to have one time unit delay. If no virtual link connecting both virtual nodes (M and N), we will conduct the third VN node embedding A with third highest node ranking value. We select the highest candidate node a from A's candidate set. After the third node embedding is done, if there exist virtual links connecting third highest virtual node A to previous two embedded virtual nodes (M and N), all links embedding will be conducted by using minimum intermediate substrate nodes preferred SP method. With respect to the remaining virtual elements (nodes and links), we still use the above embedding strategy, until the VN node and link embeddings done. We then turn to another VN embedding. Time complexity of this embedding strategy (NAL) is less than $O(|N^s_{whole}||N^V|)$, derived from [20].

It is apparent that embedding time complexity of *LANRVA-VNE* is mainly determined by our efficient node ranking approach and NAL (node and link) embedding strategy. Embedding time complexity of node ranking approach relies on calculating all candidate $NRV^s_{m,M}$ values, proved to be completed in polynomial time [20]. In addition, the NAL is able to execute the Intra-Domain node and link embedding in polynomial time, using minimum intermediate substrate nodes preferred SP approach. Thus, our *LANRVA-VNE* enables to embed the requested VN among **M**ultiple **D**istributed SNs in polynomial time, without being trapped in high computation time.

On this basis, we will not evaluate our *LANRVA-VNE* algorithm in the discrete time event for evaluating the ability of batching embedding VNs one time. Instead, we decide to evaluate our *LANRVA-VNE* in continuous time (Section IV), representing as an event of dynamic VN network service implementation.

IV. MAIN EVALUATION RESULTS

We illustrate main evaluation results in this section. We intend to highlight the efficiency of our *LANRVA-VNE* algorithm by comparing with typical two-stage heuristic embedding algorithms.

A. EVALUATION ENVIRONMENT SETTINGS

As NV is not being promoted to real network environment [2], we will adopt the synthetic network topologies to evaluate the proposed algorithms. Similar to [16]–[18], we generate 4 geographically distributed SNs to constitute the whole underlying substrate network for conducting our evaluation work. Each isolated SN is generated by using the conventional Waxman approach. In addition, each isolated SN is set to have infinite communication bandwidth connecting each other. For simplicity, we list out SN settings for each isolated SN in Table 3. With respect to each VN request, it is also generated by the Waxman model. VNs are assumed to be proposed obeying the usual Poisson distribution. We further set VN arrival rate 4 every 1000 time units in this paper [12]. With respect to each VN, we set an exponentially distributed lifetime. The average lifetime is 1000 time units. In Table 4, we also provide main settings of each VN request, such as virtual link propagation delay demand.

TABLE 3. SN and VN evaluation settings.

Substrate Nodes per SN	60
CPU, Storage, Bandwidth of Substrate	[50,100], Uniform Distribution
Each Substrate Link Progapation	One Time Unit
Substrate Node Location	[0,200], Uniform Distribution, on x and y coordinates
Number of Virtual Nodes	[2,10], Uniform Distribution
Virtual CPU, Node Storage, Link Bandwidth	[1,20], Uniform Distribution
Required Virtual Link Progapation	[1,5] Time Units
Required Virtual Node Location	[0,200], Uniform Distribution, x and y coordinates
Allowed Maximum Node Distance	[1,6], Uniform Distribution, An Integer
Allowed Maximum Intermediate Nodes	[1,3], Uniform Distribution

In addition, our evaluation runs for 100000 time units, accepted as a continuous time network event. We aim at evaluating all algorithms' embedding abilities in the long term. Therefore, 400 VN requests on average will be embedded, ensuring stable embedding results. With respect to weight factors in Expressions 4 and 5, they are set 1. α in Expression 14 is set 1. The value of factor d is set 0.15, same to the value set in [20]. With respect to the value of δ, it is set 0.0001 for iterative calculation. Researchers can reproduce our evaluation work based on these settings.

B. SELECTED ALGORITHMS FOR EVALUATION

Besides of our *LANRVA-VNE*, we select another three typical two-stage heuristic algorithms for performance evaluation (see Table 4). All selected algorithms are typical in two-stage embedding algorithm area. With respect to the *ViNE-SP* [8], it is the typical two-stage embedding algorithm, using relaxted optimization theory approach. To the *RW-SP* algorithm [11], it is another two-stage embedding algorithm using local resource (CPU and bandwidth) product value based *Markov* model. With respect to the *VNE-NTANRC-S* [12] algorithm, it is the latest two-stage embedding algorithm, quantifying multiple topology attributes (except for link interference) and network resources (except for CPU). To fit into the evaluation environment (e.g. optimizing the allowed intermediate nodes, embedding the VN among multiple distributed SNs), we make the modification work. Part codes of the modification work are available in [22].

TABLE 4. Selected algorithms for comparison.

Notation	Description
ViNE-SP [8]	Relaxed LP Node and SP Link Algorithm
RW-SP [11]	Random Walk Node and SP Link Embedding Algorithm
VNE-NTANRC-S [12]	Network Topology Attribute and Network Resource-Considered Mapping Algorithm-Stable
LANRVA-VNE	Location Aware and Node Ranking Value Assisted Embedding Algorithm

C. OUR DYNAMIC EMBEDDING STRATEGY FOR THE EVALUATION

As VNE promises to be conducted for future dynamic VN service application (Section I), it is essential to propose an embedding strategy [7] for real-time and dynamic embedding environment. Therefore, we propose an effective dynamic embedding strategy (Fig. 2), adopted by all compared embedding algorithms in this paper.

In the first place, we divide the total time units (100000 time units in this paper) into multiple continuous time windows (T_1, T_2 ... T_N). Each time window accounts for 1000 time units in this paper. In each time window, X isolated VNs (X = 1, 2, 3...), requested by multiple end users, usually come and require to be embedded onto the underlying distributed SNs. We have set X 4 in this paper (Section IV-A). Each VN will stay for an interval time (also called as VN waiting time) before expiry. By using this strategy, we set average VN lifetime equal to its allowed maximum waiting time (1000 time units). Each VN is processed sequentially according to its arriving time t_a. Previous heuristic embedding algorithms (e.g. [11]) processed VNs according to their embedding revenues (Expression 4). The VN having higher embedding revenue is usually preferred. We embed one VN request at a time. If the VN is embedded successfully, an amount of substrate resources will be occupied. We will update the SN resource information after embedding the VN. All its occupied substrate resources will be released when the VN request expires. Otherwise, the VN will be stored and waits for being re-embedded.

Note that the re-embedding of the failed VN is conducted after remaining VNs in the same time window are processed. In addition, we just conduct the re-embedding of the failed VN one time. If the re-embedding of the VN fails again, we will reject the VN. We will not serve the VN, requested by the end user, any more. We plot another figure (Fig. 2) for readers for better understanding our dynamic and real-time embedding strategy. In addition, we set one time unit one minute in this paper. Researchers and readers can reproduce our evaluation work based on these evaluation settings.

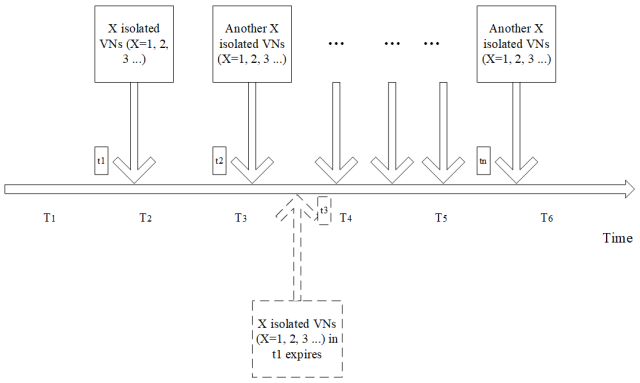


FIGURE 2. The Overview of Our Dynamic Embedding Strategy for Evaluation.

D. MAIN EVALUATION RESULTS

We illustrate the evaluation work in this subsection. The evaluation work consists of two parts. One part is to prove that our *LANRVA-VNE* is a polynomial-time algorithm. The other part intends to validate the efficiency of our *LANRVA-VNE*. Main evaluation results are plotted. For instance, Fig. 3 refers to the average revenue to cost ratio results. We plot the Fig. 4 to illustrate VN acceptance ratio results.

1) HAVING COMPETITIVE EMBEDDING EXECUTION TIME

We make Table 5 to record the embedding execution time of our selected algorithms. As introduced in subsection A, the simulation work is run on a Window 8 Desktop, with

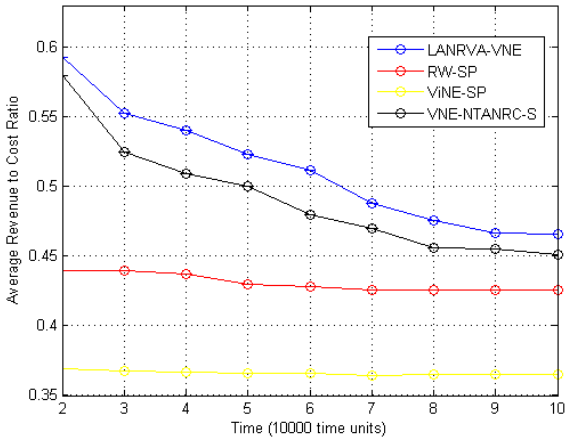


FIGURE 3. Average Revenue to Cost Ratio.

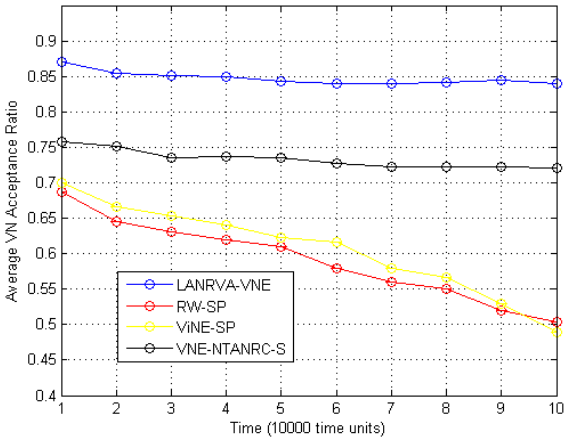


FIGURE 4. Average Acceptance Ratio.

TABLE 5. Embedding execution time comparison.

Selected Algorithms	Embedding Execution Time
ViNE-SP [8]	33.4 milliseconds
RW-SP [11]	34.3 milliseconds
VNE-NTANRC-S [12]	36.6 milliseconds
LANRVA-VNE	35.4 milliseconds

an Intel Core CPU i7-4790 3.6GHz Processor and 16.00G RAM Machine. Underlying isolated SNs and their parameters are same to what are set in subsection A. With respect to the requested VN, the number of virtual nodes in requested VN is fixed 10 (maximum size). Other parameters are same to what are set in subsection A, too (Table 3). These evaluation settings are typical in VNE research [5].

Derived from Table 5, the embedding execution time of *ViNE-SP* is 33.4 milliseconds. With respect to the *RW-SP*, the embedding execution time is 34.3 milliseconds. To the *VNE-NTANRC-S* algorithm, it has an embedding execution time of 36.6 milliseconds, whilst our proposed *LANRVA-VNE* has the embedding execution time of 35.4 milliseconds. Among the four selected algorithms, the *VNE-NTANRC-S* algorithm has the highest embedding execution time. It lies in two main reasons: one is the complex and comprehensive node ranking approach, the other is the separated node and link embeddings.

Therefore, we can draw the conclusion that our proposed *LANRVA-VNE* can conduct the VN embedding in polynomial time. This conclusion is in accordance with the embedding time complexity discussion in Section III. Compared with selected algorithms (*ViNE-SP* and *RW-SP*), our *LANRVA-VNE* needs extra VN embedding execution time to embed the VN. However, extra embedding time is not much. In addition, the embedding quality can be greatly improved, illustrated in the following subsection. Compared with the *VNE-NTANRC-S*, our *LANRVA-VNE* costs fewer embedding time. Therefore, our *LANRVA-VNE* algorithm is competitive and promises to be promoted to future dynamic VNE environment.

2) HAVING HIGHEST AVERAGE REVENUE TO COST RATIO

In VNE research, embedding revenue and embedding cost metrics are widely accepted as the main performance metrics for evaluating various VN embedding algorithms. However, it is meaningless to evaluate embedding revenue and cost separately. Based on previous research [5], [16], the embedding revenue to cost is widely accepted as another effective performance metric, revealing the physical resource usage efficiency. Higher is the value of ratio, more efficient is the embedding algorithm. Therefore, we plot the average revenue to cost ratio results of four compared algorithms in Fig. 3.

Derived from Fig. 3, we can apparently make the conclusion that our *LANRVA-VNE* achieves highest revenue to cost ratio among all four algorithms. This behavior lies in two main reasons. First of all, it is the one embedding stage of our *LANRVA-VNE*. Compared embedding algorithms (*ViNE-SP*, *RW-SP* and *VNE-NTANRC-S*) conduct each VN embedding in two embedding stages. No or part coordination between node and link embeddings will lead to unnecessary underlying resources consumption. Thus, it is better to propose one-stage embedding algorithm in order to optimize the ratio performance. Secondly, it lies in the constraint of limiting maximum allowed intermediate nodes. Optimizing this constraint enables to save extra embedding cost of our *LANRVA-VNE*. Therefore, with respect to revenue to cost ratio performance metric, our *LANRVA-VNE* performs the best among all compared algorithms.

In addition, we can discover the fact that the ratios of all four algorithms decay with evaluation time expanding. This discover lies in the fact there are no infinite underlying resources (CPUs, node storages and communication bandwidths) for accommodating more and more requested VNs. With more VNs coming, more extra underlying resources will be demanded. Therefore, the average revenue to cost ratios decrease.

3) AN IMPROVEMENT OF 10% VN ACCEPTANCE RATIO

In Fig. 4, we illustrate the average VN acceptance ratio performance of all four algorithms. In VNE, this metric is used to evaluate any embedding algorithm's embedding ability. Higher VN acceptance ratio, stronger embedding ability. Apparently, our *LANRVA-VNE* algorithm performs best among the four algorithms. For instance, since 20000 time point, the acceptance ratio of *LANRVA-VNE* is approaching 0.853. Meanwhile, the VN acceptance ratio of *VNE-NTANRC-S* is 0.75. This is more than 10% lower than our *LANRVA-VNE*. This behavior runs as expected. It lies in that our *LANRVA-VNE* enables to embed nodes and links per VN in one embedding. This embedding scheme maximizes the node and link embedding coordination. Thus, extra underlying resources will be reserved for accepting following VNs. With respect to remaining three algorithms (*ViNE-SP*, *RW-SP* and *VNE-NTANRC-S*), their inefficient embedding schemes lead to low acceptance, comparing with our one-stage embedding algorithm.

4) HIGHEST NODE STORAGE UTILIZATION AMONG ALL ALGORITHMS

In Fig. 5, we plot the average node storage utilization of all compared algorithms. With the evaluation time extending, more VNs are required to be embedded. As a consequence, accepting more proposed VNs will contribute to the increase of node storage utilization. Through conducting the evaluation, we can easily find that the node storage utilizations of all selected algorithms increase.

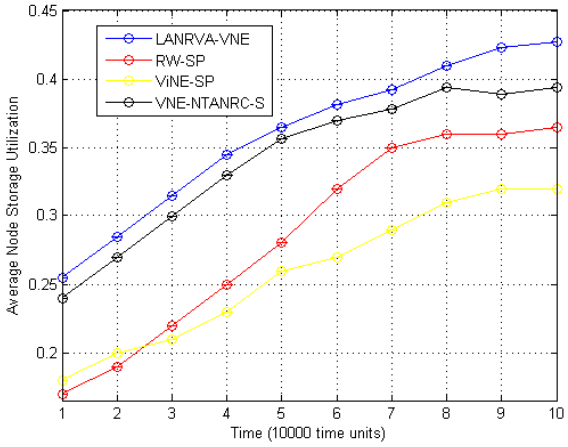


FIGURE 5. Average Node Storage Utilization.

With respect to the average node storage utilization (Fig. 5), our *LANRVA-VNE* has an advantage over the selected two-stage embedding algorithms, especially to *RW-SP* and *ViNE-SP* algorithms. The advantage is very apparent. The cause of our *LANRVA-VNE* having a higher node utilization is the fact that our *LANRVA-VNE*'s embedding ability is stronger than the selected *RW-SP* and *ViNE-SP* algorithms. Through quantifying more topology attributes and calculating node values in an iterative manner, our *LANRVA-VNE* explores the substrate nodes' embedding abilities accurately and deeply. Therefore, our *LANRVA-VNE* accepts more requested VNs and has a higher node storage utilization. Our *LANRVA-VNE* will further embed VNs onto substrate nodes efficiently until no remaining underlying nodes. In addition, our *LANRVA-VNE* will load remaining substrate nodes to their full capacities. Compared with the efficient heuristic *VNE-NTANRC-S*, our *LANRVA-VNE* is able to conduct the embedding in one embedding stage. Thus coordinating and scheduling the VN embedding more efficiently. Consequently, more extra node resources will be reserved. With time going by, the node storage utilization of our *LANRVA-VNE* will behave better than the *VNE-NTANRC-S*, as plotted in the Fig. 5.

5) HAVING A HIGHER COMMUNICATION BANDWIDTH UTILIZATION

In Fig. 6, we plot the average communication bandwidth utilizations of all selected embedding algorithms. In VNE research, the average communication bandwidth utilization is another important metric for evaluating the link resource usage. Derived from the Fig. 6, we can find the average communication bandwidth utilizations of all four algorithms increase with the evaluation time passing by. It runs as expected because more and more VNs are processed. More link resources are consumed. Thus, the average communication bandwidth utilization increases.

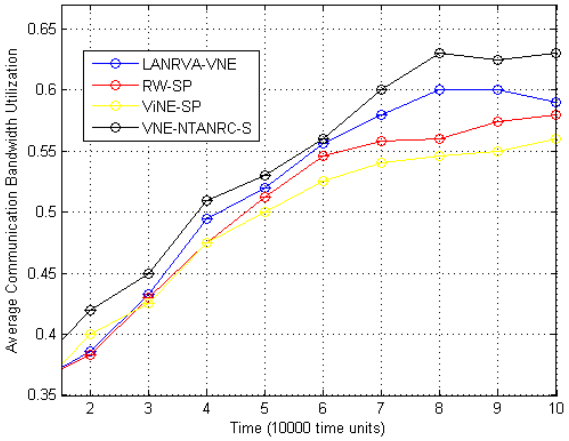


FIGURE 6. Average Communication Bandwidth Utilization.

In addition, the evaluation results of communication bandwidth utilization (Fig. 6) have the same behavior for all compared VNE algorithms, demonstrated in Fig. 5. All four embedding algorithms behave similarly. It is owing to the fact that the well known shortest path (SP) approach is used to deal with the virtual link embedding. No path splitting action is allowed in our evaluation. Therefore, link utilizations of all algorithms, no matter one-stage or two-stage related, run similarly. However, some differences exist. For instance, the bandwidth utilization of our *LANRVA-VNE* is a bit lower than the *VNE-NTANRC-S*.

V. CONCLUSION

To deal with **M**ultiple **D**istributed VN embedding problem, we propose a **L**ocation **A**ware and **N**ode **R**anking **V**alue **A**ssisted embedding algorithm *LANRVA-VNE*. The proposed *LANRVA-VNE* is able to embed each given VN in one embedding stage. In addition, the *LANRVA-VNE* conducts the VN embedding among multiple distributed SNs in polynomial time. Main evaluation results reveal that our *LANRVA-VNE* algorithm can significantly improve more than 10% VN acceptance ratio than typical two-stage embedding algorithms (e.g. *VNE-NTANRC-S*, *RW-SP* and *ViNE-SP*). Other metrics (e.g. average revenue to cost ratio) are also illustrated to highlight our *LANRVA-VNE*. In the future research, we will further evaluate the performance of our *LANRVA-VNE* algorithm, with no or little exposed underlying resource information.

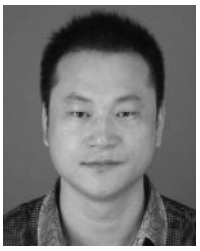
ACKNOWLEDGMENT

The conference version of this journal paper was submitted to the 2019 IEEE International Conference on Communications for possible acceptance in Oct. 15th, 2018.

REFERENCES

[1] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*, 5th ed. London, U.K.: Pearson, 2011.
[2] "Cisco visual networking index: Global mobile data traffic forecast update, 2015–2020," Cisco, San Jose, CA, USA, White Paper, 2016.
[3] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, Apr. 2015.
[4] (2018), *China Tower*. [Online]. Available: http://zgtt.dlzb.com/
[5] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 4th Quarter, 2013.
[6] H. Cao, L. Yang, Z. Liu, and M. Wu, "Exact solutions of VNE: A survey," *China Commun.*, vol. 13, no. 6, pp. 48–62, Jun. 2016.
[7] H. Cao, H. Hu, Z. Qu, and L. Yang, "Heuristic solutions of virtual network embedding: A survey," *China Commun.*, vol. 15, no. 3, pp. 186–219, Mar. 2018.
[8] P. Zhang, H. Yao, C. Qiu, and Y. Liu, "Virtual network embedding using node multiple metrics based on simplified ELECTRE method," *IEEE Access*, vol. 6, pp. 37314–37327, 2018.
[9] M. Melo, S. Sargento, U. Killat, A. Timm-Giel, and J. Carapinha, "Optimal virtual network embedding: Node-link formulation," *IEEE Trans. Netw. Service Manag.*, vol. 10, no. 4, pp. 356–368, Dec. 2013.
[10] A. Jarray and A. Karmouch, "Decomposition approaches for virtual network embedding with one-shot node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 1012–1025, Jun. 2015.
[11] P. Zhang, H. Yao, and Y. Liu, "Virtual network embedding based on the degree and clustering coefficient information," *IEEE Access*, vol. 4, pp. 8572–8580, 2016.
[12] H. Cao, L. Yang, and H. Zhu, "Novel node-ranking approach and multiple topology attributes-based embedding algorithm for single-domain virtual network embedding," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 108–120, Feb. 2018.
[13] H. Cao, Y. Zhu, G. Zheng, and L. Yang, "A novel optimal mapping algorithm with less computational complexity for virtual network embedding," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 1, pp. 356–371, Mar. 2018.
[14] O. Soualah, N. Aitsaadi, and I. Fajjari, "A novel reactive survivable virtual network embedding scheme based on game theory," *IEEE Trans. Netw. Service Manage.*, vol. 14, no. 3, pp. 569–585, Sep. 2017.
[15] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proc. 1st ACM Workshop VISA*, 2009, pp. 81–88.
[16] I. Houidi, W. Louati, W. B. Ameur, and D. Zeghlache, "Virtual network provisioning across multiple substrate networks," *Comput. Netw.*, vol. 55, no. 4, pp. 1011–1023, 2011.
[17] D. Dietrich, A. Rizk, and P. Papadimitriou, "Multi-provider virtual network embedding with limited information disclosure," *IEEE Trans. Netw. Service Manage.*, vol. 12, no. 2, pp. 188–201, Jun. 2015.
[18] T. Mano, T. Inoue, D. Ikarashi, K. Hamada, K. Mizutani, and O. Akashi, "Efficient virtual network optimization across multiple domains without revealing private information," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 477–488, Sep. 2016.
[19] M. Newman, *Networks: An Introduction*. Oxford, U.K.: Oxford Univ. Press, 2010.
[20] T. H. Cormen, C. Stein, R. Rivest, and C. Leiserson, *Introduction to Algorithms*, 2nd ed. New York, NY, USA: McGraw-Hill, 2001.
[21] G. Golub and C. Van Loan, *Matrix Computations* (Johns Hopkins studies in Mathematical Sciences), 3rd ed. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 1996.
[22] (2016). *Simulation Platform for Scotfield Cao*. [Online]. Available: http://pan.baidu.com/s/1gekPZrl

HAOTONG CAO (S'17) received the B.S. degree in communication engineering from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2015, where he is currently pursuing the Ph.D. degree. He was a Visiting Scholar with Loughborough University, U.K., in 2017. He has published multiple IEEE transaction/journal/magazine papers, since 2016. His research interests include next-generation network, wireless communication theory, and resource allocation in wired and wireless networks. He has served as the TPC Member for multiple IEEE conferences, such as the IEEE INFOCOM, IEEE ICC, IEEE GLOBECOM, IEEE WCNC, and IEEE VTC. He has received the 2018 Postgraduate National Scholarship of China. He is also serving as the Reviewer for multiple academic journals, such as the IEEE Internet of Things Journal, the IEEE/ACM Transactions on Networking, the IEEE Transactions on Network and Service Management, and *Computer Networks* (Elsevier).



YONGAN GUO (M'18) received the Ph.D. degree from the Nanjing University of Posts and Telecommunications (NJUPT), Nanjing, China, in 2018. He is currently a Lecturer with the College of Telecommunications and Information Engineering, NJUPT. His main research areas include wireless communication technology, Internet of Things technology and applications, and network virtualization.

YUE HU received the B.S. degree in network engineering and the M.S. degree in electronic and communication engineering from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2013 and 2016, respectively, and the M.S. degree in communication engineering from Melbourne University, Melbourne, Australia, in 2015. She is currently with China Mobile Communications Group Jiangsu Co., Ltd. Her research interests include Internet of Things technology and future network architecture.

SHENGCHEN WU (S'18) received the B.S. degree in communication engineering from the Nanjing University of Posts and Telecommunications (NJUPT), Nanjing, China, in 2012, and the M.S. degree in electric computer engineering from New York University, USA, in 2014. He is currently pursuing the Ph.D. degree with the College of Telecommunications and Information Engineering, NJUPT. His main research areas include wireless communication technology and Internet of Things technology and applications.

HONGBO ZHU received the B.S. degree in communications engineering from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 1982, and the Ph.D. degree in information and communications engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 1996. He is currently a Professor and was the Vice President with the Nanjing University of Posts and Telecommunications. He is also the Head of the Coordination Innovative Center of IoT Technology and Application, Jiangsu, which is the first government authorized Coordination Innovative Center of the IoT in China. He has authored or co-authored over 300 technical papers published in various journals and conferences. He is currently leading a big group and multiple funds on the IoT and wireless communications with a current focus on architecture and enabling technologies for the Internet of Things. His research interests include the Internet of Things, mobile communications, and wireless communication theory. He also serves as a referee or an expert in multiple national organizations and committees.

LONGXIANG YANG is currently a Full Professor and the Doctoral Supervisor with the College of Telecommunications and Information Engineering, Nanjing University of Posts and Telecommunications (NJUPT), Nanjing, China, where he is also the Vice Dean. He has fulfilled multiple National Natural Science Foundation projects of China. He has authored or co-authored over 200 technical papers published in various journals and conferences. His research interests include cooperative communication, network coding, wireless communication theory, key technologies of 5G mobile communication systems, ubiquitous networks, and the Internet of Things.