Collaborative Attributes and Resources for Single-Stage Virtual Network Mapping in Network Virtualization

Haotong Cao, Hongbo Zhu, and Longxiang Yang

Abstract: **Virtual network embedding (VNE) is the virtualized node and link resources allocation problem in network virtualization environment, aiming at achieving the simultaneous optimal node and link mapping assignment per VN. Currently, a large number of mapping algorithms for VNE exist in the academia. Existing algorithms mostly focus on mapping each virtual network (VN) in two ordered stages: First virtual node mapping stage and second virtual link mapping stage, leading to non-optimal VN mapping assignment. Though multiple one-stage algorithms exist, adopting either optimization theory or graph theory, they usually involve much more VN assignment calculation time. Hence, these one-stage algorithms are not suitable for dynamic network scenario. Not to mention practical VNE application. Based on the background and our gained research results, we propose another heuristic mapping algorithm *VNE-ARS*, completing all nodes and links per VN in one single embedding stage. Each VN embedding assignment can be completed in polynomial time, using our *VNE-ARS*. Main network attributes and resources are collaborated, serving as the mapping criterion of our *VNE-ARS*. In order to highlight our *VNE-ARS* strength, we conduct the evaluation experiments. We compare our *VNE-ARS* algorithm against existing mapping algorithms.**

Index Terms: **Heuristic algorithm, mapping algorithm, network attributes, one single stage, two ordered stages, virtual network embedding.**

I. INTRODUCTION

Future network based on virtualization technology has been emphasized [1] as the evolutionary direction of next generation network in the research community. In the future network virtualization environment, one key technical issue is virtualized node and link resources allocation. This issue is called as virtual network embedding (VNE) [2], struggling to achieve the simultaneous optimal node and link mapping assignment per VN. To deal with VNE, a large number of mapping algorithms are studied. Corresponding, review papers on VN embedding algorithms have been published [3]–[5] in recent years. In VNE research, two terms, embedding and mapping, can be used interchangeably.

However, most of existing mapping algorithms [4], [6]–[14] embed the given VN in two ordered and isolated stages: first virtual node mapping stage and second virtual link mapping stage. This mapping strategy cannot guarantee the optimal or suboptimal embedding assignment per VN in many cases. Instead, only a feasible VN embedding assignment can be achieved. Following this strategy, physical node and link resources cannot be consumed efficiently in the long term. Not to mention optimizing VN acceptance performance. Hence, there is a great demand of developing one-stage mapping algorithms that integrate the VN mapping in one single stage. Though one-stage mapping algorithms [3] have been well developed in recent years, originated from optimization theory [15], [16] or graph theory [17], [18], another key flaw exists. The key flaw is that this type of one-stage algorithms have high embedding assignment calculation time per VN. As VNE is designed for future dynamic network scenario, it is important to minimize the mapping assignment calculation time and guarantee the embedding quality.

Based on our gained knowledge and recent research results, we propose another single-stage heuristic mapping algorithm in this paper. In our proposed algorithm, main existing network attributes and resources [19] are collaborated and quantified as the mapping criterion for sorting nodes. We conduct the VN embedding in one single stage, with the assistance of the mapping criterion. By extracting these characteristics, our algorithm is labeled as *VNE-ARS*. The *VNE-ARS* has the goals of maximizing VN acceptance ratio and using physical node and link resources to their full capacities. Different from previous one-stage algorithms [3], our *VNE-ARS* is able to calculate the VN embedding assignment within polynomial time. That is to say, our *VNE-ARS* has a lower VN embedding assignment calculation time than existing one-stage algorithms [15]–[18]. In addition, the VN embedding quality of our *VNE-ARS* improves, comparing with previous typical two-stages algorithms. Therefore, our *VNE-ARS* is suitable for future virtualized resources application. To further validate our *VNE-ARS* efficiency, we conduct the evaluation experiments. For example, our *VNE-ARS* has an apparent advantage over its two-stages version, in terms of VN acceptance ratio results.

We list out main contributions of our paper below:

- A single-stage heuristic embedding algorithm, *VNE-ARS*, is proposed in this paper. The *VNE-ARS* integrates isolated node embedding and link mapping per VN in one procedure. In addition, the VN embedding assignment can be calculated in

Manuscript received June 27, 2019; approved for publication by Hongjian Sun, Division III, July 28, 2019.

Our research work is supported by the National Key Research and Development Program of China under Grant 2018YFC1314903, the National Natural Science Foundation of China under Grant 61372124 and 61427801.

H. Cao, H. Zhu, and L. Yang are with Jiangsu Key Laboratory of Wireless Communications, Nanjing University of Posts and Telecommunications, Nanjing 210003, China, and also with the Key Laboratory of Broadband Wireless Communication and Sensor Network Technique, Ministry of Education, Nanjing University of Posts and Telecommunications, Nanjing 210003, China, email: {1015010309, zhuhb, yanglx}@njupt.edu.cn.

L. Yang is the corresponding author.

Digital Object Identifier: 10.1109/JCN.2019.000045

polynomial time, using our *VNE-ARS*. Therefore, *VNE-ARS* promises to be applied to future dynamic network scenario.

- In our *VNE-ARS*, the number of implemented virtual constraints is five: CPU, node storage, node location, node deployment time and link bandwidth. While in previous VNE research [2], [5], only CPU, node location and bandwidth are implemented as virtual constraints for each requested VN.
- A comprehensive evaluation is conducted in order to highlight our *VNE-ARS* strength. The evaluation part consists of three sub-parts: comparing with *VNE-ARS*'s sub-algorithms sub-part, comparing with *VNE-ARS*'s two-stages version subpart, and comparing with existing two-stages heuristic algorithms sub-part.

Remaining sections are organized as follows. Related work is discussed in Section II. Network model and evaluation metrics are presented in Section III. In Section IV, our *VNE-ARS* is detailed. Evaluation work is conducted in Section V. At last, conclusion marks are made.

II. RELATED WORK

In this section, we briefly look back on existing two-stages algorithms and latest one-stage mapping algorithms.

With respect to existing two-stages mapping algorithms in the literature, they usually embed each demanded VN in two ordered stages: first virtual node embedding stage and second virtual link embedding stage. Greedy node mapping strategy and shortest path (SP) link mapping strategy are adopted for virtual node embedding and virtual link embedding, respectively [4], [5]. Both mapping stages can be completed in polynomial time. Hence, these algorithms can be evaluated in dynamic network scenario. Serving as the basis of VN embedding, these two-stages algorithms have their self-developed node ranking approaches so as to sort all nodes [5] ahead, no matter physical or virtual. For example, Gong *et al.* [6] adopted the node degree [19], node strength and Markov Random Walk model to calculate all node values in an iterative manner. Feng *et al.* [7] incorporated multiple node attributes to calculate node values in a direct way. In [8], Cheng *et al.* [8] adopted Google page ranking model to calculate node and its adjacent link values in an iterative manner. Cao *et al.* [9], [11] added link interference attribute for calculate node values. In [10], node degree and clustering coefficient attributes were assisted to rank nodes, too. Other node sorting approaches, used in [12]–[14], were not detailed in this paper. In general, efficient node sorting approach has been widely accepted as a vital role of two-stages mapping algorithms. Hence, it is important to collaborate network attributes and resources in order to reveal node importance before conducting embedding.

With respect to the one-stage type, multiple algorithms have been proposed over the past six years. We detail the latest one-stage algorithms. In [15], Cao *et al.* adopted the pure integer linear programming (ILP) model, integrating virtual nodes and links embeddings in one single stage. However, the pure ILP model-based embedding strategy was too time consuming to calculate VN embedding assignment in polynomial time. Hence, Cao *et al.* constructed candidate sets for limiting the number of variables and evaluated the proposed algorithm in small scaled network scenario. While in [16], Li *et al.* adopted the ILP model to deal with different types of VN requests. The highlight of [16] was the category method of VN request. However, the ILP-based algorithm proposed by Li could not be evaluated in dynamic network scenario. In [17], Lischka *et al.* used the subgraph isomorphism approach of graph theory to embed VN in one embedding stage. Subgraph isomorphism approach based one-stage algorithm guaranteed VN embedding quality. However, the embedding assignment calculation time per VN grew exponentially with the size of network expanding. Gong *et al.* [18] adopted the compatibility graph of graph theory to embed each VN in one embedding stage. However, this algorithm was usually trapped in high calculation time. Generally, existing one-stage algorithms cannot be evaluated in dynamic network scenario. Though ensuring the quality of VN embedding assignment, one-stage algorithms [3] are time consuming.

Therefore, we propose another heuristic mapping algorithm *VNE-ARS* that integrates two ordered mappings into one single stage. Our *VNE-ARS* is able to calculate VN embedding assignment within polynomial time. Hence, our *VNE-ARS* can be promoted to future dynamic network scenario. Based on our gained knowledge and previous research results [5], we collaborate main network attributes and resources in our *VNE-ARS*, serving as the mapping criterion. In addition, we implement five different virtual constraints while adopting our *VNE-ARS* to embed VNs. In the existing publications [4], [5], at most three constraints, CPU, node location and link bandwidth, are executed. To further strengthen our *VNE-ARS*, we conduct the comprehensive experiments. For example, evaluation results reveal that our *VNE-ARS* outperforms all selected two-stages mapping algorithms.

III. NETWORK MODEL AND EVALUATION METRICS

A. Network Model

The network model for virtualization research consists of two sub-models: One is the substrate network model and the other is the virtual network (request) model.

The substrate network (SN) can be modeled as an undirected graph $G^s(V^s, E^s)$, where V^s and E^s represent the sets of substrate nodes and substrate links in the network, respectively. $|V^s|$ and $|E^s|$ record the number of substrate nodes and substrate links, respectively. $v^s \epsilon V^s$ represents a substrate node in V^s. Each v^s has its CPU $CPU(v^s)$, node storage $Sto(v^s)$, node deployment time $DeT(v^s)$ and geographical location $Loc(v^s)$. Each $e^s \epsilon E^s$ has the bandwidth $bw(e^s)$ for transmitting data flow. In addition, P^s denotes the set of loop-free substrate paths in $G^s(V^s, E^s)$. $P^s_{v^s_{|M|}, v^s_{|N|}}$ denotes the set of loop-free paths between certain two substrate nodes $v^s_{|M|}$ and $v^s_{|N|}$ in $G^s(V^s, E^s)$. $|M|$ and $|N|$ are certain two different integers between [1, $|V^s|$]. Note that ρ_{CPU}, ρ_{Sto} and ρ_{bw} denote the weights of CPU, node storage and link bandwidth. CPU, node storage and link bandwidth are selected as network resources that will be used in Section IV of this paper.

The virtual network (request) model can be modeled as an undirected graph $G^v(V^v, E^v, T^v)$, where V^v and E^v are the sets of all virtual nodes and virtual links in $G^v(V^v, E^v, T^v)$, re-

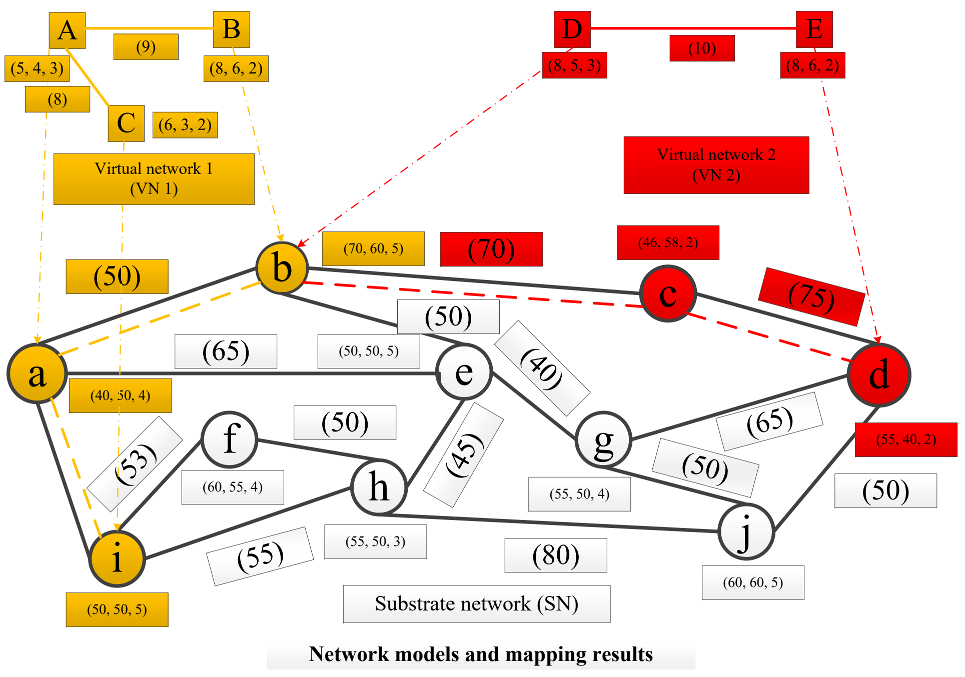

Fig. 1. Network models and mapping results.

spectively. T^v denotes the time attributes of $G^v(V^v, E^v, T^v)$: arriving time point $ATP(G^v)$, expiring time point $ETP(G^v)$, maximum allowed waiting time $MAWT(G^v)$, embedding execution time $EET(G^v)$. Each virtual node $v^v \epsilon V^v$ has its required CPU $CPU(v^v)$, demanded node storage $Sto(v^v)$, maximum allowed deployment time $MADT(v^v)$ and requested geographical location $Loc(v^v)$. $r(v^v)$ is the coverage radius of virtual node v^v. Each virtual link $e^v \epsilon E^v$ has its bandwidth demand $bw(e^v)$ for communication. Following the introduction of two network models, we will describe the VN embedding that consists of virtual nodes mapping and virtual links embedding. As usual, all virtual nodes of the VN are firstly mapped. Then, all virtual links of the VN are embedded. VNs are processed one by one. When one VN is being mapped, other VNs must be waiting. For better understanding VN embedding, we take one virtual network (request), labeled as $G^v(V^v, E^v, T^v)$, as an example.

First all virtual nodes mapping: In the virtual node mapping stage, each virtual node in the $G^v(V^v, E^v, T^v)$ is mapped by the unique function $Func_N(\,)$:$V^v \to V^s$. We take one virtual node $v^v_{|M|}$ in $G^v(V^v, E^v, T^v)$ as an example.

$$\begin{aligned} &Func_N(v^v_{|M|}) \in V^s \\ &Func_N(v^v_{|M|}) = Func_N(v^v_{|N|}), \\ &if\ and\ only\ if\ v^v_{|M|} = v^v_{|N|} \end{aligned}$$

s. t.

$$CPU(v^v_{|M|}) \leq CPU(Func_N(v^v_{|M|})) \tag{1}$$

$$Sto(v^v_{|M|}) \leq Sto(Func_N(v^v_{|M|})) \tag{2}$$

$$MADT(v^v_{|M|}) \geq DeT(Func_N(v^v_{|M|})) \tag{3}$$

$$Dis(Loc(v^v_{|M|}), Loc(Func_N(v^v_{|M|}))) \leq r(v^v_{|M|}), \tag{4}$$

where (1) means that the CPU demand of $v^v_{|M|}$ must be fulfilled by the mapped $Func_N(v^v_{|M|})$. To (2), it indicates that demanded node storage of $v^v_{|M|}$ must be fulfilled by the mapped $Func_N(v^v_{|M|})$. To (3), it means that node deployment time of the mapped $Func_N(v^v_{|M|})$ must be within the required deployment time of $v^v_{|M|}$. To (4), it indicates that the Euclidean distance between $v^v_{|M|}$ and $Func_N(v^v_{|M|})$ must be within the coverage radius of virtual node $r(v^v_{|M|})$. From (1) to (4), all four equations must be fulfilled. Otherwise, the virtual node $v^v_{|M|}$ is not mapped successfully.

Second all virtual links mapping: After completing the virtual node mapping successfully, it is the second virtual link mapping of $G^v(V^v, E^v, T^v)$. Path splitting and path interference are not considered in this paper. That is to say, each virtual link e^v is mapped onto one physical path directly. We take one virtual link $e^v_{|M|,|N|}$, connecting virtual nodes $v^v_{|M|}$ and $v^v_{|N|}$, as an example. The link virtual embedding of $G^v(V^v, E^v, T^v)$ is performed by the unique virtual link function $Func_L(\,) : E^v \to P^s$.

$$Func_L(e^v_{|M|,|N|}) \subseteq P^s_{Func_N(v^v_{|M|})Func_N(v^v_{|N|})}$$

s. t.

$$bw(e^v_{|M|,|N|}) \leq bw(P^s_{Func_N(v^v_{|M|})Func_N(v^v_{|N|})}), \tag{5}$$

where (5) indicates that the communication bandwidth demand of $e^v_{|M|,|N|}$ must be fulfilled by the selected substrate path $P^s_{Func_N(v^v_{|M|})Func_N(v^v_{|N|})}$. With respect to remaining virtual links in the $G^v(V^v, E^v, T^v)$, their communication demands must be fulfilled, using the link mapping function. Otherwise, reject the $G^v(V^v, E^v, T^v)$.

As presented above, it is the embedding description of VN $G^v(V^v, E^v, T^v)$. For readers to have a better understanding, we plot Fig. 1, presenting two VNs embedding onto a shared SN. VN 1 is marked in yellow while VN 2 is marked in red. Underlying SN is marked in white while occupied nodes are marked in yellow or red. Three numbers in each rectangle indicate CPU, node storage and node deployment time. One number in each rectangle represents the link bandwidth. As shown in Fig. 1, different nodes in the same VN must be embedded onto different substrate nodes: $A \to a$, $B \to b$ and $C \to i$ for VN 1, $D \to b$ and $E \to d$ for VN 2. In addition, different virtual nodes belonging to different VNs are allowed to share the same substrate node. For example, virtual node B of VN 1 and virtual node D of VN 2 can share same physical node b.

B. Main Evaluation Metrics

This sub-section includes main performance metrics for evaluating VNE algorithms: VN embedding revenue, embedding cost, VN acceptance ratio, node resource utilization and link resource utilization.

At first, it is the VN embedding revenue. For telecommunication providers, VN embedding revenue represents the revenue by implementing the VN G^v. We formulate (6).

$$Rev(G^v_{T^v_{ser}}) = \begin{cases} Rev(G^v) \cdot T^v_{ser},\ if\ G^v\ is\ embedded \\ 0, \qquad\qquad else, \end{cases} \tag{6}$$

where T^v_{ser} is the VN G^v service time. $T^v_{ser} = ETP(G^v) - ATP(G^v) - ECT(G^v) - EET(G^v)$. Take note that $ECT(G^v)$ represents the embedding assignment calculation time of G^v. Other time terms have been introduced in Section III-A. In addition, the sum of $ECT(G^v)$ and $EET(G^v)$ must not be more than $MAWT(G^v)$. The per-unit revenue $Rev(G^v)$ of (6) is formulated in (7). CPU, node storage and communication bandwidth are selected. Weights method is adopted to balance different network resources.

$$Rev(G^v) = \rho_{CPU} \cdot \sum_{v^v \in V^v} CPU(v^v) + \rho_{Sto} \cdot \sum_{v^v \in V^v} Sto(v^v) + \rho_{bw} \cdot \sum_{e^v_{|M|,|N|} \in E^v} bw(e^v_{|M|,|N|}) \tag{7}$$

Secondly, it is the VN embedding cost metric, indicating the amount of consumed physical resources for implementing the VN G^v. We formulate (8).

$$Cost(G^v_{T^v_{ser}}) = \begin{cases} Cost(G^v) \cdot T^v_{ser},\ if\ G^v\ embedded \\ 0, \qquad\qquad else, \end{cases} \tag{8}$$

where $Cost(G^v)$ represents the per-unit cost of G^v. Other parameters used in (8) are same to parameters used in (6). We further formulate (9) to detail the per-unit cost of G^v.

$$Cost(G^v) = \rho_{CPU} \cdot \sum_{v^v \in V^v} CPU(v^v) + \rho_{Sto} \cdot \sum_{v^v \in V^v} Sto(v^v) + \rho_{bw} \cdot \sum_{e^v_{|M|,|N|} \in E^v} \sum_{p^s \in P^s} num^{p^s}_{e^v_{|M|,|N|}} \cdot bw(e^v_{|M|,|N|}), \tag{9}$$

where $num^{p^s}_{e^v_{|M|,|N|}}$ records the number of substrate links in the substrate path that is used to accommodate the virtual link $e^v_{|M|,|N|}$.

Afterwards, it is the VN acceptance ratio metric. This metric is designed for quantifying any mapping algorithm's embedding ability. The VN acceptance ratio is determined by the number of accepted VNs and total requested VNs during the evaluation time. See (10).

$$VNAcce = AcceptedVNs/TotalVNs \tag{10}$$

Next, it is the node resource utilization metric. To the type of node resources, CPU and node storage are considered in this paper. Hence, we formulate (11).

$$NodeUtil = ConsumedNode/TotalNode, \tag{11}$$

where $ConsumedNode$ refers to the total consumed CPU or node storage of certain substrate node (e.g. v^s) after embedding all VNs in the evaluation time. $TotalNode$ refers to the initial amount of CPU or node storage of certain substrate node (e.g. v^s). While in this paper, we illustrate CPU utilization for illustration (Section V).

At last, it is the link resource utilization metric. Only link bandwidth is considered in this paper. Its equation is similar to (11). For saving space, we do not formulate the link resource metric in this paper.

IV. *VNE-ARS* ALGORITHM

Our *VNE-ARS* algorithm is detailed. Firstly, we collaborate main selected network attributes and resources. Secondly, we introduce a convenient node sorting approach, serving as the mapping criterion of all nodes, including the substrate and the virtual. Finally, we detail the single-stage VN mapping.

A. Collaborated Attributes and Resources

In previous publications [3]–[5], researchers usually had their self-developed node ranking approaches. By using their own node approaches, calculated node values could serve as the mapping criterion. In each node ranking approach, certain network attributes and network resources were incorporated. Based on these backgrounds and our gained knowledge [9], [11], we select multiple main attributes and resource. We use one SN $G^s(V^s, E^s)$ in order to formulate main attributes and resources. Certain substrate nodes $v^s_{|M|}$ and $v^s_{|N|}$ belong to $G^s(V^s, E^s)$. $|M|$ and $|N|$ are certain two different integers between [1, $|V^s|$].

Firstly, we talk about the selected network resources. Selected resources in this paper are CPU, node storage and link bandwidth, that have been mentioned in Section III-A.

Secondly, we talk about the selected network attributes. Selected network attributes are extracted from various previous node ranking approaches [5], proved efficient and effective before:

The first network attribute is the node degree attribute ($Degree(v^s_{|M|})$). In network theory, degree attribute of certain node $v^s_{|M|}$ records the number of direct links connecting $v^s_{|M|}$ to its adjacent nodes, revealing the node connectivity in the network. Higher node degree, more connectivity of the node.

The second network attribute is the node strength attribute ($Strength(v^s_{|M|})$). To certain node $v^s_{|M|}$, its strength attribute indicates the weights sum of its adjacent links. In this paper, link bandwidths represent the weights over links. That is to say, if the node strength is higher, the node has more adjacent resources for mapping VNs.

The third network attribute is the node closeness ($Closeness(v^s_{|M|})$). This node attribute is about recording the node distance in the network. To certain node $v^s_{|M|}$, it is usually connected to remaining nodes in the network. Hence, it is essential to record the total distances from node $v^s_{|M|}$ to remaining nodes. See (12).

$$Closeness(v^s_{|M|}) = \sum_{v^s_{|M|}, v^s_{|N|} \in N^s, |M| \neq |N|} Dis(Loc(v^s_{|M|}), Loc(v^s_{|N|})) \tag{12}$$

The fourth node attribute is the node centrality ($Centrality(v^s_{|M|})$). This node attribute indicates the number of times that the selected node $v^s_{|M|}$ serves as a bridge within the shortest path connecting certain two different nodes (e.g. $v^s_{|P|}$ and $v^s_{|Q|}$). This attribute can reveal the contribution of node $v^s_{|M|}$ to measuring the network connectivity. We formulate (13).

$$Centrality(v^s_{|M|}) = \sum_{v^s_{|M|}, v^s_{|P|}, v^s_{|Q|} \in E^s, v^s_{|M|} \neq v^s_{|P|} \neq v^s_{|Q|}} \frac{\mathit{Number}(v^s_{|P|}, v^s_{|Q|})(v^s_{|M|})}{\mathit{Number}(v^s_{|P|}, v^s_{|P|})}, \tag{13}$$

where $\mathit{Number}(v^s_{|P|}, v^s_{|Q|})(v^s_{|M|})$ represents the number of shortest paths connecting node $v^s_{|P|}$ and node $v^s_{|Q|}$, passing node $v^s_{|M|}$. $\mathit{Number}(v^s_{|P|}, v^s_{|Q|})$ represents the number of shortest paths connecting node $v^s_{|P|}$ and node $v^s_{|Q|}$.

B. Convenient Node Sorting Approach

By collaborating above network attributes and resources, we propose a convenient node approach. The calculated node values can serve as the mapping criterion of the following one single-stage mapping. Procedures of the node approach are presented:

Firstly, we define and formulate a class of network attribute block, labeling as *NAB*. The *NAB* class has three different subclasses: *NAB-TWO*, *NAB-THREE* and *NAB-FOUR*, aiming at quantifying network attributes. See (14)–(16). Main differences of three sub-classes are selected network attributes. Node $v^s_{|M|}$ is selected as an example.

$$NAB - TWO(v^s_{|M|}) = Degree(v^s_{|M|}) \cdot Strength(v^s_{|M|}) \tag{14}$$

$$NAB - THREE(v^s_{|M|}) = Degree(v^s_{|M|}) \cdot Strength(v^s_{|M|}) \cdot Closeness(v^s_{|M|}) \tag{15}$$

$$NAB - FOUR(v^s_{|M|}) = Degree(v^s_{|M|}) \cdot Strength(v^s_{|M|}) \cdot Closeness(v^s_{|M|}) \cdot Centrality(v^s_{|M|}) \tag{16}$$

Secondly, we intend to define and formulate another metric, named as network resource block *NRB*, quantifying the considered network resources (CPU, node storage and link bandwidth) and functional attributes (location and node deployment time). Stimulated from the known *Coulomb's law* in electromagnetism area and the *Newton's law* in gravitational field, we formulate (17). Node $v^s_{|M|}$ is selected as an example.

$$NRB(v^s_{|M|}) = \alpha \cdot \sum_{v^s_{|M|}, v^s_{|N|} \in N^s} \frac{CPU(v^s_{|M|}) \cdot CPU(v^s_{|M|}) \cdot bw(e^s_{|M|,|N|})}{Dis(Loc(v^s_{|M|}), Loc(v^s_{|N|}))^2}, \tag{17}$$

where α is a constant. In addition, at least one path connects $v^s_{|M|}$ and $v^s_{|N|}$. Otherwise, the value of (17) is 0.

Thirdly, it is the procedure of calculating node sorting value, labeled as *NSV* in this paper. Derived from our published survey [5], the direct product of *NAB* and *NRB* can be used to represent the node value of node $v^s_{|M|}$, $NSV(v^s_{|M|})$ (18). However, in many extreme network cases, such as sparse networks [5], [20], direct product cannot reveal the node embedding ability accurately, leading to inefficient substrate resources utilization and low VN embedding acceptance [5]. Hence, another calculation method should be proposed.

$$NSV(v^s_{|M|}) = NAB(v^s_{|M|}) \cdot NRB(v^s_{|M|}) \tag{18}$$

$$NSV(v^s_{|M|})\% = \frac{NSA(v^s_{|M|})}{\sum_{v^s_M \in V^s} NSA(v^s_{|M|})} \tag{19}$$

Derived from *Markov* random model [21], we decide to calculate the accurate $NSV(v^s_{|M|})$ in order to be the mapping criterion of VN embedding. As three sub-classes exist in *NAB*, there should exist three different $NSV(v^s_{|M|})$ values. For simplicity, we just talk about one $NSV(v^s_{|M|})$. Firstly, we compute the initial node sorting value percentages in (19). We use certain node $v^s_{|M|}$ as an example in (19). Then, we constitute an initial node embedding ability vector, labeled as $NSVVec^0$ in this paper (20). In addition, we define two kinds of transformation probabilities in (21) and (22): Attribute transformation probability and resource transformation probability.

$$NSVVec^0 = (NSV(v^s_1)\%, \cdots, NSV(v^s_{|V^s|})\%)^T \tag{20}$$

$$Pro_{NAB(v^s_{|M|}, v^s_{|N|})} = \frac{NAB(v^s_{|N|})}{\sum_{v^s_{|M|} \in V^s} NAB(v^s_{|M|})} \tag{21}$$

$$Pro_{NRB(v^s_{|M|}, v^s_{|N|})} = \frac{NRB(v^s_{|N|})}{\sum_{v^s_{|M|} \in V^s} NRB(v^s_{|M|})}, \tag{22}$$

where $v^s_{|M|}$ and $v^s_{|N|}$ are any two different nodes in the whole network. With using the *Markov* model approach, we can calculate the eventual node values of all nodes. With respect to node $v^s_{|M|}$, its node value is shown in (23) by $(k+1)$ rounds of calculation.

$$NSV(v^s_{|M|})^{k+1} = (1-d) \cdot \sum_{v^s_{|M|} \neq v^s_{|N|}, v^s_{|N|} \in V^s}$$

Algorithm 1 A convenient node calculation.

```
Require: Network $G^s(V^s, E^s)$, a small positive value $\delta$
Ensure: Vector $NSVVec$
1: Get matrix $M$, the initial $NSVVec^0$
2: Define an iteration number $k$, $k = 0$.
3: Define a variable $\varepsilon$, is $\infty$.
4: while $\varepsilon \geq \delta$ do
5:     $NSVVec^{k+1}$ =M$\cdot NSVVec^k$;
6:     $\varepsilon = \|NSVVec^{k+1} - NSVVec^k\|$;
7:     $k = k + 1$;
8: end while
9: $NSVVec = NSVVec^{k+1}$
```

$$Pro_{NAB(v^s_{|N|}, v^s_{|M|})} \cdot NSV(v^s_{|M|})^k$$

$$+d \cdot \sum_{m \neq n, n \in N} Pro_{NRB(v^s_{|N|}, v^s_{|M|})} \cdot NSV(v^s_{|M|})^k, \quad (23)$$

where d is the damping factor. The factor is within the range of (0,1). In order to express all node sorting values in the form of a vector $NSVVec$, expression 24 is formulated.

$$NSVVec^{k+1} = (1-d) \cdot M_1 \cdot NSVV^k +$$

$$+d \cdot M_2 \cdot NSVV^k, \quad (24)$$

where M_1 and M_2 are the transition matrices of attribute probability and resource probability. Both matrices have $(|N| \cdot |N|)$ dimensions. Equation (24) can be further calculated in (25).

$$NSVVec^{k+1} = [(1-d) \cdot M_1 + d \cdot M_2] \cdot NSVVec^k$$

$$= M \cdot NSVVec^k, \quad (25)$$

where M is defined as the chain transition matrix. We can easily get the fact that the eigenvalue of M is not more than 1 [21]. Consequently, the conclusion that traffic matrix M is the stable matrix is true [20]. Therefore, the $NSVVec^{k+1}$ will eventually converge to a stable vector.

Though it is easy to get the vector converged, it is difficult to calculate the final vector of (25). The time complexity of directly calculating (25) is approaching $O(|V|^3)$. Time complexity grows exponentially with the network scale expanding. Hence, it is impractical to calculate directly. Instead, a convenient iterative approach is adopted. Through k iterations, the vector will be converged [20]. Time complexity of the calculation is decreased to $O((|V|) \cdot log(1/\delta))$. δ is a small positive number to control iteration times. See Algorithm 1.

C. One Single-Stage Mapping Strategy

In this subsection, we detail the one single-stage mapping strategy of our *VNE-ARS*.

Using the above conventional node approach, we can calculate the SN and VN nodes values. Then we store all substrate node values and all virtual node values in separated substrate set and virtual set, respectively. All calculated node values are regarded as the mapping criterion. Afterwards, we backup all resource and topology information of the SN. Next, we conduct two virtual nodes' greedy embeddings. Both virtual nodes have the first two highest node sorting values of the VN. Note that the embedded substrate nodes must reserve enough network resources (CPU, storage, allowed location and node deployment time) so as to fulfill two highest virtual nodes constraints. For instance, if the virtual node capacity cannot be fulfilled by all substrate nodes of the SN, we will reject the VN. Consequently, the SN information will not be updated. If two highest virtual nodes are embedded, we will conduct the subsequent link embedding of two highest virtual nodes. If there exists a virtual link connecting two highest virtual nodes, the virtual link will be simultaneously mapped by using the SP approach. If no virtual link connecting both nodes, we will turn to mapping the third virtual node with the third highest node value. After the third node embedding completes, if there exist virtual links connecting third highest virtual node to previous two embedded virtual nodes, all links embedding will be conducted by using minimum intermediate nodes preferred SP method. With respect to remaining virtual elements, we repeat the above embedding strategy.

Until all virtual nodes and links are successfully mapped, we will output the VN embedding results. This VN mapping is completed within one stage. Time complexity of the one single-stage mapping is less than $O(|V^s||V^v|)$ [21]. We need to take note that with certain two connected virtual nodes embedded, their simultaneous virtual link embedding starts. We will point out some key points of our simultaneous virtual link embedding. Firstly, to embed the virtual link optimally, we adopt minimum intermediate nodes preferred SP method, aims at consuming less bandwidth resources. Secondly, we conduct one pruning procedures in the simultaneous virtual link embedding. We try to delete all substrate links having no available bandwidths for accommodating virtual link. Time complexity of SP method is less than $O(|E^s||E^v|log|V^s|)$.

D. Algorithm Complexity of VNE-ARS

The algorithm complexity of *VNE-ARS* is determined by the node values calculation and one single-stage mapping. The complexity of calculating node values lies in the iterative-based calculation. The calculation can be completed in polynomial time [21]. The complexity of single stage mapping is less than the sum of two isolated stages (greedy node mapping and SP link mapping) that can be completed in polynomial time [20]. Hence, our *VNE-ARS* is a polynomial-time VN embedding algorithm.

V. EVALUATION EXPERIMENTS

A. Experiment Settings

Since virtualization research is still in its infancy [1], prototypes for evaluating VNE have not been fully developed. We conduct the evaluation experiments to validate our *VNE-ARS* algorithm. We generate the underlying SN, using GT-ITM tool. The number of nodes is set to be 100. Each node has a possibility of 0.5 to connect remaining nodes in the SN. With respect to the node resources (CPU, node storage), they are integers, following the uniform distribution [50, 100]. To each physical node location, it is uniformly distributed in the (200*200) two-dimensional plane. To each physical node deployment time, it

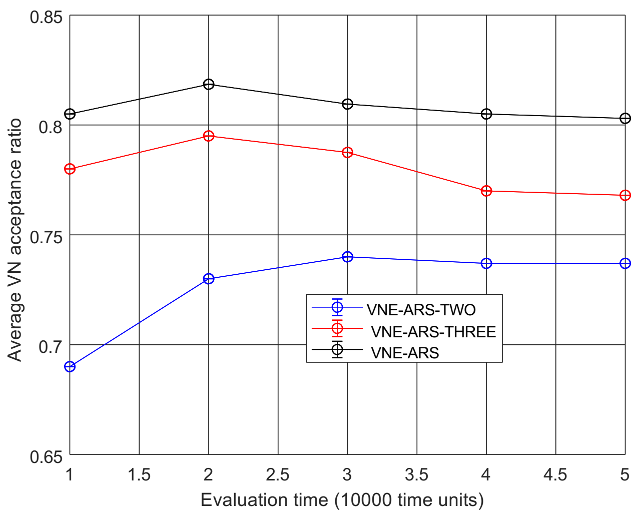

Fig. 2. Average VN acceptance ratio (80 substrate nodes).

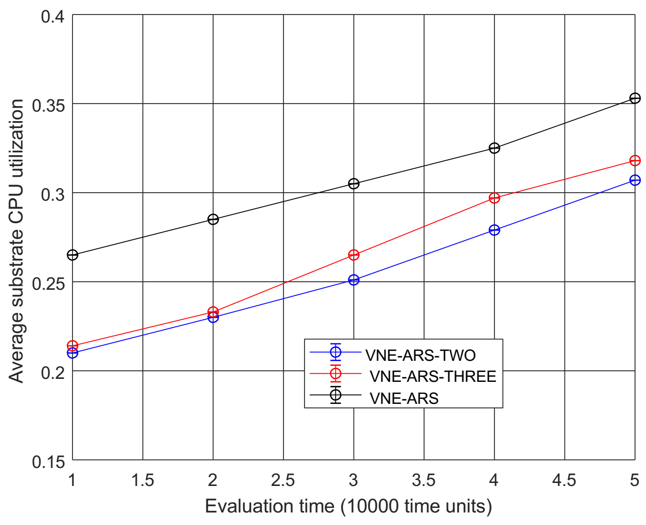

Fig. 3. Average substrate CPU utilization (80 substrate nodes).

is set to be one millisecond. With respect to the link communication bandwidths, they are integers, following the uniform distribution [50, 100]. To the requested VNs, they are generated by using GT-ITM. VNs are requested following the Poisson distribution, up to 5 every 100 time units. An exponentially distributed VN lifetime is set to be 20 time units. With respect to virtual resource demands (CPU, storage, bandwidth), they are integers, following the uniform distribution [1, 20]. Each virtual node location is uniformly distributed in the (200*200) plane. The coverage radius of each virtual node is uniform distributed within [5, 10]. For each virtual node deployment time demand, it is set to be one millisecond.

We run the experiments up to 100000 time units. That is to say, approaching 50000 VNs will be processed. With respect to resource factors in (7), they are set to be 1. We make our evaluation codes, available in [22], [23]. Take note that 1 time unit represents 1 minute. We introduce settings in detail for following researchers to re-produce our experiments easily.

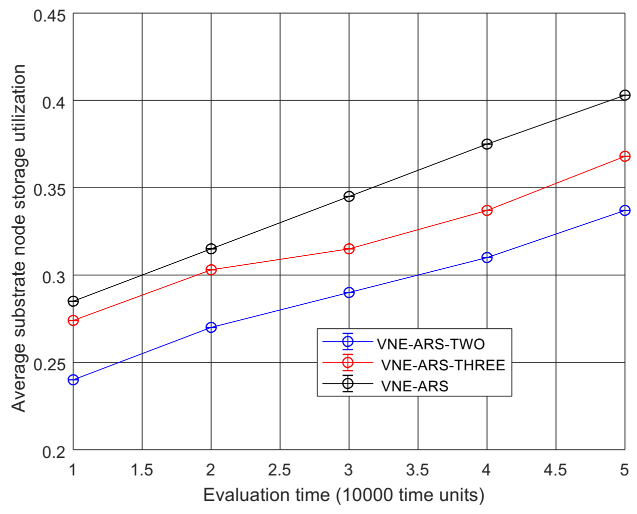

Fig. 4. Average substrate node storage utilization (80 substrate nodes).

B. Comparison Between VNE-ARS and Its Sub-Algorithms

As three sub-classes of *NAB* exist (Section IV-B), our conventional node approach consists of three different sub-approaches strictly. Consequently, three different classes of node values for mapping can be calculated. Therefore, three algorithms exist: *VNE-ARS-TWO*, *VNE-ARS-THREE* and *VNE-ARS-FOUR*. As *VNE-ARS-FOUR* considers all four network attributes, it can be regarded as *VNE-ARS*. In this sub-section, we conduct a performance comparison between *VNE-ARS* and its sub-algorithms (*VNE-ARS-TWO* and *VNE-ARS-THREE*), aiming at validating the effects of network attributes.

With respect to the evaluation settings, they are same to what are presented in Section V-A. Two main differences exist: the SN network scale and evaluation time. The SN network scaled is set to be 80. The evaluation time is set to be 50000 time units. Ranging from Fig. 2 to Fig. 5, we plot main evaluation results.

In Fig. 2, VN acceptance ratios of *VNE-ARS* and its two sub-algorithms are plotted. Derived from Fig. 2, two main conclusions can be made. The first conclusion is that there are finite physical resources for accommodating VN requests. We can see the fact that all three algorithms will decrease to stable ratios around 40000 time unit. The second conclusion is that our *VNE-ARS*, collaborating all four main attributes and network resources, perform best among all three algorithms. With incorporating the node centrality attribute in our node sorting approach, it does benefit to improving embedding quality. Hence, more resource space will be reserved. More VNs can be accepted.

Ranging from Fig. 3 to Fig. 5, evaluation results of node and link resource utilizations (CPU, node storage and link bandwidth) are plotted. Derived from all three figures, our *VNE-ARS* consumes the most amount of resources. These behaviors support the VN acceptance ratio of our *VNE-ARS* directly.

In general, our *VNE-ARS*, considering the most number of attributes and resources, performs better than its two sub-algorithms. In another word, collaborating more attributes do benefits to improving embedding quality.

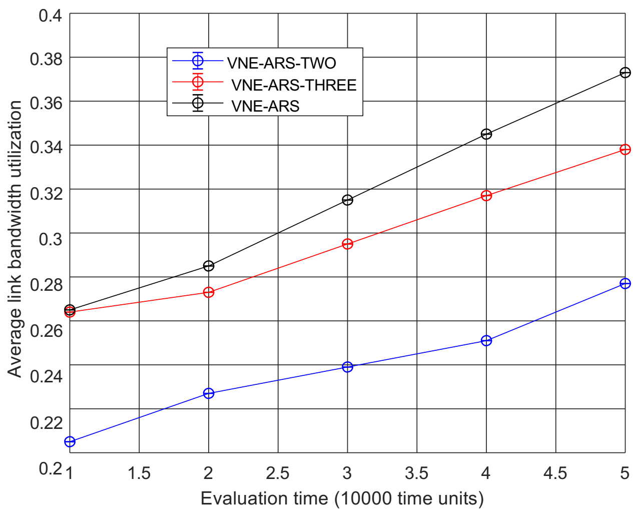

Fig. 5. Average link bandwidth utilization (80 substrate nodes).

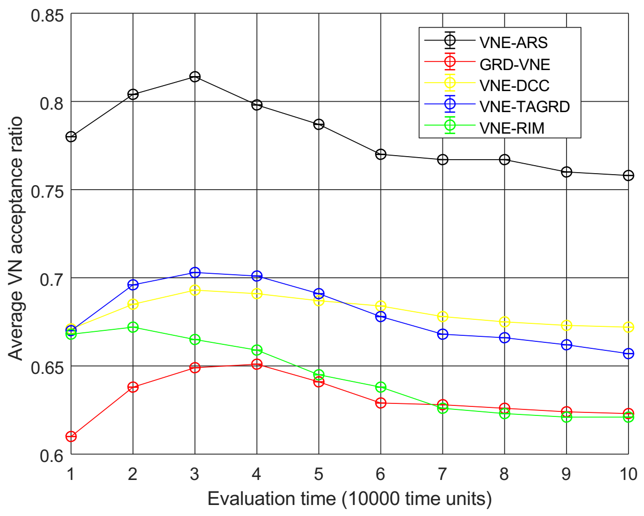

Fig. 7. Average VN acceptance ratio (100 substrate nodes).

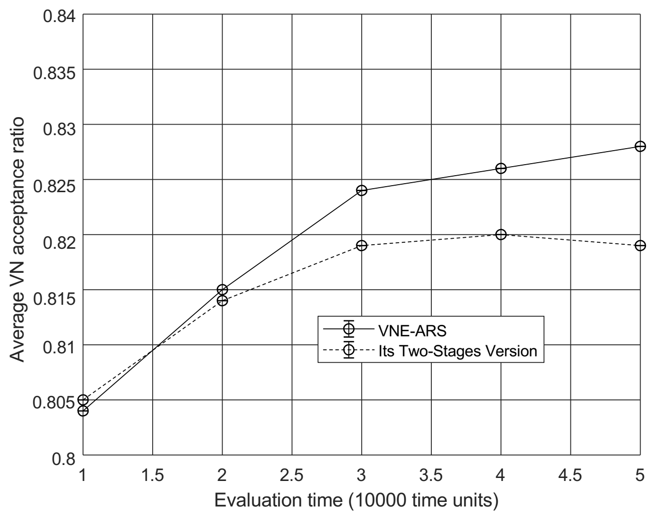

Fig. 6. Average VN acceptance ratio (100 substrate nodes).

C. *Comparison Between VNE-ARS and Its Two-Stages Version*

In this sub-section, we conduct another comparison between *VNE-ARS* and its two-stages version. With respect to the two-stages version of our *VNE-ARS*, it refers to conduct the VN embedding in two isolated stages, though using the same node sorting approach. Due to limited pages, we just plot the VN acceptance ratio results.

In Fig. 6, we can find that two algorithm run similarly in the early stage of evaluation time. With time extending to around 20000 time unit, our *VNE-ARS* performs better than its two-stages version. The performance advantage of our *VNE-ARS* lies in its single-stage mapping, saving extra physical resources. Hence, more physical resources are reserved for following VNs. With time passing by, more and more VNs can be accommodated by our *VNE-ARS*. Eventually, our *VNE-ARS* (0.827) has higher VN acceptance ratio than its two-stages version (0.818).

D. *Comparison Between VNE-ARS and Typical Two-Stages Algorithms*

By adopting the settings in sub-section A, we conduct a performance comparison between our *VNE-ARS* and typical two-stages algorithms. Selected two-stages algorithms are either representative or latest in the literature. Selected algorithms are *GRD-VNE* [6], *VNE-DCC* [10], *VNE-TAGRD* [11] and *VNE-RIM* [12].

VN Acceptance Ratio Performance Comparison: As shown in Fig. 7, we conduct the VN acceptance ratio comparison of all five embedding algorithms. Two apparent conclusions can be made easily. With respect to the first conclusion, it is the finite substrate resources for accommodating VNs. We can discover that all algorithms increase in the first place. Then their acceptance ratios will decrease. Their acceptance ratios will achieve the dynamic balance in the end. With respect to the second conclusion, it is the performance advantage of our *VNE-ARS*. Since 90000 time unit, VN acceptance ratio of our *VNE-ARS* will remain around 0.77. With respect to the best behaved two-stages algorithm, it is around 0.67 (*VNE-DCC*), which is 10% lower than our *VNE-ARS*. Two main reasons are responsible for our *VNE-ARS* advantage: the node sorting approach collaborating main network attributes and resources and the combined one single-stage mapping strategy.

VN Embedding Revenue and Embedding Cost Performance Comparison: In Figs. 8 and 9, we plot the VN embedding revenue and embedding cost results of all selected algorithms, respectively. It is not comprehensive to discuss the VN embedding revenue or embedding cost results separately. Two conclusions can be made form both figures. With respect to the first conclusion, it is the discovery that VN embedding revenue results can reveal the VN acceptance ratio results. As usual, more and more VN demands are accommodated, more embedding revenues can be earned. As the SN has finite physical resources, VN acceptance will decrease to the stable state. Correspondingly, VN embedding revenues will achieve to a balance. With respect to the second conclusion, it is the fact that VN embedding cost of all

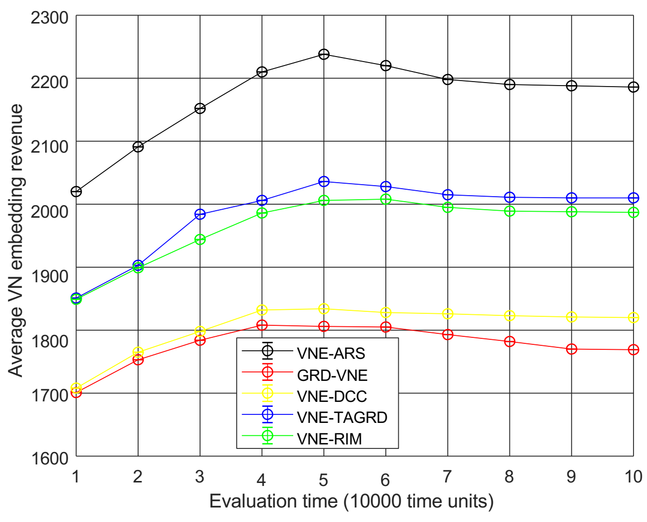

Fig. 8. Average VN embedding revenue (100 substrate nodes).

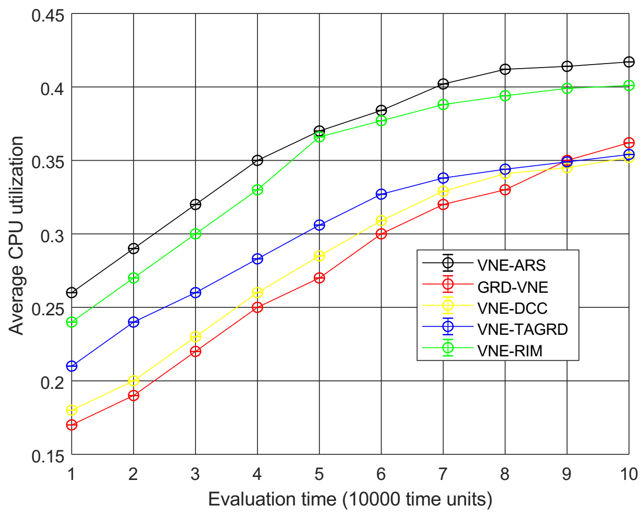

Fig. 10. Average CPU utilization (100 substrate nodes).

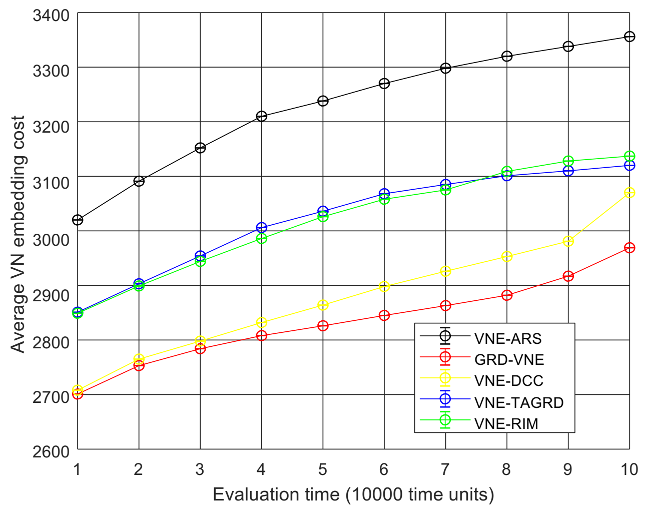

Fig. 9. Average VN embedding cost (100 substrate nodes).

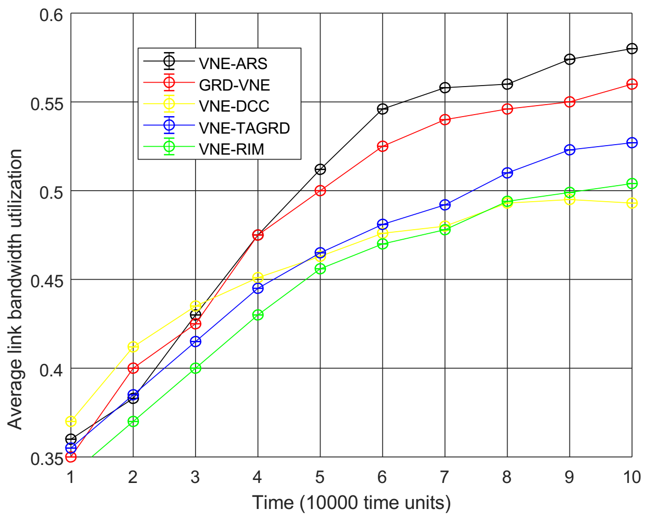

Fig. 11. Average link bandwidth utilization (100 substrate nodes).

selected algorithms increase with time extending. This conclusion lies in the fact that more and more VNs are requested with evaluation going. Therefore, more and more substrate resources will be consumed. The SN will load all its resources to its full capacity.

CPU and Bandwidth Utilization Performance Comparison: CPU utilization and link bandwidth utilization results are plotted in Figs. 10 and 11, respectively. In Fig. 10, we can find that CPU utilizations of all selected algorithms increase, especially to our *VNE-ARS*. In general, our *VNE-ARS* consumes the most amount of node CPU resources. Owing to the collaborated network attributes and resources, our *VNE-ARS* can ensure efficient VNs mapping. Hence, the underlying substrate resources will be fully consumed. To the link bandwidth results, we can find the fact that the link bandwidth results of all algorithms run similarly. It is owing to the reason that the same link mapping strategy (SP approach) is used by all algorithms.

E. Discussion of Evaluation Experiments

We briefly talk about the embedding quality of our *VNE-ARS* algorithm in this sub-section.

In sub-section B, we compare our *VNE-ARS* with its sub-algorithms that incorporate part network attributes. By conducting the evaluation experiments, our *VNE-ARS* proves to have stronger embedding ability than its sub-algorithms. Hence, it is necessary to consider more network attributes. Only node degree and strength are not enough to guarantee VN embedding quality.

In sub-section C, we conduct the evaluation work in order to highlight the advantage of single-stage embedding. If adopting the two-stages mapping strategy, the lack of node and link coordination will lead to inefficient VN mapping and extra substrate resources consumption. In the end, VN acceptance ratio will decrease. Hence, it is necessary to adopt the one-stage mapping strategy.

In sub-section D, we compare our *VNE-ARS* with multiple two-stages algorithms. With respect to the two-stages algorithms, they embed each VN in two isolated stages: first node embedding and second link embedding. They aim at finding a feasible VN mapping assignment in order to minimize embedding assignment calculation time. Hence, the VN mapping quality can not be guaranteed. While in VNE, VN embedding quality and VN embedding calculation time cannot be achieved simultaneously. On these backgrounds, we make the compromise between VN embedding quality and VN embedding calculation time. We propose our *VNE-ARS* algorithm. On the one hand, we adopt the heuristics in order to save calculation time. On the other hand, we conduct one single-stage VN mapping so as to ensure better embedding quality. Evaluation results validate our *VNE-ARS* algorithm efficiency.

VI. CONCLUSION MARKS

We propose one single-stage heuristic mapping algorithm *VNE-ARS* in this paper. Our *VNE-ARS* enables to calculate each VN mapping assignment within polynomial time. We evaluate our *VNE-ARS* in a comprehensive manner in order to highlight its efficiency. Evaluation results reveal that our *VNE-ARS* outperforms not only existing two-stages mapping algorithms, but also its two-stages version. For further research, we intend to evaluate our *VNE-ARS* by implementing the prototypes.

REFERENCES

[1] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: A survey," *IEEE Commun. Mag.*, vol. 51, no. 11, pp. 24–31, 2013.
[2] A. Belbekkouche, Md. Mahmud Hasan, and A. Karmouch, "Resource discovery and allocation in network virtualization," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 4, pp. 1114–1128, Feb. 2012.
[3] H. Cao, L. Yang, Z. Liu, and M. Wu, "Exact solutions of VNE: A survey," *China Commun.*, vol. 13, no. 6, pp. 48–62, June 2016.
[4] A. Fischer, J. Botero, M. Till Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 4th Quart. 2013.
[5] 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.
[6] L. Gong, Y. Wen , Z. Zhu, and T. Lee, "Revenue-driven virtual network embedding based on global resource information," in *Proc. IEEE GLOBECOM*, Dec. 2013, pp. 2294–2299.
[7] M. Feng, J. Liao, J. Wang, S. Qing, and Q. Qi, "Topology-aware virtual network embedding based on multiple characteristics," in *Proc. IEEE ICC*, June 2014, pp. 2956–2962.
[8] X. Cheng *et al.*, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 38–47, Apr. 2011.
[9] 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 IoT J.*, vol. 5, no. 1, pp. 108–120, Feb. 2018.
[10] 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.
[11] H. Cao, Y. Zhu, L. Yang, and G. Zheng, "A efficient mapping algorithm with novel node- ranking approach for embedding virtual network," *IEEE Access*, vol. 5, pp. 22054–22066, 2017.
[12] M. Lu, Y. Lian, Y. Chen, and M. Li, "Collaborative dynamic virtual network embedding algorithm based on resource importance measures," *IEEE Access*, vol. 6, no. 1, pp. 55026–55042, 2018.
[13] G. Sun, H. Yu, V. Anand, and Y. Li, "A cost efficient framework and algorithm for embedding dynamic virtual network requests," Fu. Gen. Com. Sys., vol. 29, no. 5, pp. 1267–1277, July 2013.
[14] J. Zhang *et al.*, "Dynamic virtual network embedding over multilayer optical networks," *J. Optical Commun. Networking*, vol. 7, no. 9, pp. 918–927, Sept. 2015.
[15] 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.
[16] Z. Li, Z. Lu, S. Deng, and X. Gao, "A self-adaptive virtual network embedding algorithm based on software-defined networks," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 1, pp. 362–373, Mar. 2019.
[17] J. Lischka and H. Karl, "A virtual network embedding algorithm based on subgraph isomorphism detection," in *Proc. ACM VISA*, Aug. 2009, pp. 81–88.
[18] L. Gong, H. Jiang, Y. Wang, and Z. Zhu, "Novel location-constrained virtual network embedding (LC-VNE) algorithms towards integrated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 24, no. 6, pp. 3648–3661, 2016.
[19] L. Freeman, The development of social network analysis. Empirical Press Vancouver, BC, 2004.
[20] T. H. Cormen, C. Stein, R. Rivest, and C. Leiserson, Introduction to Algorithms, 2nd ed. McGraw-Hill Higher Education, 2001.
[21] G. Golub and C. van Van Loan, Matrix Computations (Johns Hopkins studies in Mathematical Sciences) (3rd Edition), Johns Hopkins University Press, Oct. 1996.
[22] H. Cao, S. Hu, and L. Yang, "New Functions added to ALEVIN for evaluating virtual network embedding," in *Proc. IEEE ICCC*, Oct. 2016, pp. 2411–2414.
[23] [Online]. *Simulation Platform for Scotfield Cao*, 2018. Available: http://pan.baidu.com/s/1gekPZrl.

Haotong Cao (S'17) received B.S. Degree in Communication Engineering from Nanjing University of Posts and Telecommunications (NJUPT) in 2015. He is currently pursuing his Ph.D. Degree in NJUPT, Nanjing, China. He was a Visiting Scholar of Loughborough University, U.K. in 2017. He has served as the TPC member of multiple IEEE conferences, such as IEEE INFOCOM, IEEE ICC, IEEE Globecom. He is also serving as the Reviewer of multiple academic journals, such as IEEE Internet of Things Journal, IEEE/ACM Transactions on Networking, IEEE Transactions on Network and Service Management and (Elsevier) Computer Networks. He has published multiple IEEE Trans./Journal/Magazine papers since 2016. His research interests include wireless communication theory, resource allocation in wired and wireless networks. He has received the 2018 Postgraduate National Scholarship of China. He has been awarded 2019 IEEE ICC SecSDN Workshop Best Paper Award.

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

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