Received: 4 January 2019 | Revised: 1 September 2019 | Accepted: 25 November 2019

DOI: 10.1002/ett.3849

RESEARCH ARTICLE

WILEY

Virtual network embedding on massive substrate networks

Chenggui Zhao[1,2] | **Behrooz Parhami**[2]

[1]School of Information, Yunnan University of Finance and Economics, Kunming, China

[2]Department of Electrical and Computer Engineering, University of California, Santa Barbara, California

Correspondence
Chenggui Zhao, School of Information, Yunnan University of Finance and Economics, Kunming 650221, China.
Email: zhaochenggui@126.com

Present Address
Chenggui Zhao, School of Information, Yunnan University of Finance and Economics, Kunming 650221, China

Funding information
National Natural Science Foundation of China, Grant/Award Number: 61562089

Abstract

To alleviate the computational burden of previous virtual network embedding (VNE) approaches when the resource network scales up significantly, we propose an efficient node ranking strategy that considers both global and local topological characteristics of the substrate network in mapping virtual nodes to physical nodes. This method ranks the substrate network nodes in two stages. First, all nodes are ranked globally with respect to the stationary distribution of the entire network. Then, a connected subset of the ranked substrate nodes, forming the H-admissible embedding subgraph, is extracted. Finally, the subgraph nodes are ranked according to a local node ranking vector derived from a random-walking scheme. The local rank vector is resolved using discrete Green's function satisfying the Dirichelet boundary condition. The more accurate association of node demands and resources that our proposed method provides leads to both better acceptance ratio and lower computational overhead. These claims have been justified via theoretical and algorithmic presentation of our scheme and offer experimental results obtained through simulation, to confirm its execution efficiency and solution quality compared with a couple of previous VNE proposals.

1 | INTRODUCTION

Network virtualization technologies are attracting much attention from the network research community. Internet service providers (ISPs) equipped with virtualization tools allocate physical network resources to provision virtual resources to users (see Table 1 for key abbreviations). This function is referred to as virtual network embedding (VNE), mapping, or provisioning.[1] The trend toward broader application of network virtualization results from the fact that it allows the sharing of physical resources among several virtual networks on a dynamic, as-needed basis and brings about the cost benefits of deploying standardized, high-volume components. Not surprisingly, the 5G communication network standard is predicated on extensive use of virtualization. Particularly, it has been reported[2] that network virtualization is a potential solution in 5G wireless networks for supporting multimedia services in a flexible and cost-effective way.

If treating VNE as the problem of node assignment or matching between two graphs, schemes emulating Google's PageRank can be exploited for dealing with VNE, as done, for example, in the work of Cheng et al.[3] Roughly speaking, current approaches in VNE research are afflicted with high computational overhead in optimization methods and low acceptance rate in heuristic algorithms.[4] Real-time response to user's requirements constitutes a main practical challenge, given that, for instance, embedding a 50-node virtual network (VN) into 125 physical nodes needs 362.25 seconds if

Chenggui Zhao, Chenggui Zhao was an academic visitor at Department of Electrical & Computer Engineering, University of California, Santa Barbara, CA 93106, during September 2016 to September 2017.
Behrooz Parhami is a Life Fellow of IEEE, a Fellow of IET and British Computer Society (e-mail: parhami@ece.ucsb.edu).

TABLE 1 Key abbreviations and acronyms

Term	Interpretation
AES	Admissible embedding subgraph
AR	Acceptance ratio
C/R	Cost/revenue ratio
ILP	Integer linear programming
ISP	Internet service provider
NUR	Node utilization ratio
SN	Substrate network
VN	Virtual network
VNE	Virtual network embedding
VNR	Virtual network request

done via DViNE-SP[5] and 23.61 seconds via RW-MM-SP.[3] Additional challenges arise from the fact that most heuristic algorithms suffer from low acceptance rates. For example, an algorithm proposed in the work of Cheng et al[3] can lead to rejected VNR due to node exhaustion in vertex cut set or link exhaustion on edge cut set, even though other peers have unused resources. Intuitively, when virtual nodes v_1 and v_2 with link demand 10 attempt to occupy substrate resource u_1 and u_2 connected only by the path $u1xyu_2$, the inability of the link (x, y) to fulfill link demand on (v_1, v_2) will cause this request to be rejected. This implies that topology of physical devices affects the acceptance rate of VNE algorithms.

Solutions of several combinatorial optimization problems,[6,7] including node ranking, can be tightly connected to eigen-solutions of graph-related matrices, such as the adjacency matrix, transition probability matrix of random walk,[8,9] and Laplacian, given that calculating the eigenpair of these matrices consumes far less time than advanced combinatorial search procedures in large objective spaces. Even greater benefits would result from the avoidance of numerical computation if one can derive a closed form solution for the Perron vector of the graph adjacency matrix, that is, the eigenvector corresponding to the maximal eigenvalue. Our motivation in advocating global and local random walking on graphs to solve the VNE problem arises from the fact that the magnitude and distribution of eigenvalues and eigenvectors of a graph substantially reflect its geometric properties and randomness characteristics.[6-8] This encourages us to exploit the Perron vector of the transition probability matrices of random walks on substrate network (SN) and VN to deduce the node and link mapping for VNE task.

The main contribution of this work is to decompose the general iterative convergence process of node ranking vector, arising from VNE algorithms that use node ranking, into two procedures: rough global ranking and detailed local ranking. For estimating global SN node ranking, this work exploits the stationary probability of classical random walk. Then, chooses a node subset from global node ranking to induce the maximal admissible embedding subgraph (AES) in SN. A candidate node for such a subset has relatively more residual resources than the remaining nodes, short distance to chosen nodes, and connection to at least one chosen node to ensure connectivity. Then, calculate the stationary probability of local random walk on induced subgraph to obtain the node ranking vector through solving the discrete Green's function.[10-12] Our presentation entails theoretical and algorithmic frameworks, in addition to simulations conducted for a couple of periods on Hub-Star network with ISP characteristics. The results demonstrate that our algorithm outperforms other algorithms, in terms of both efficiency and quality, with respect to multiple assessment metrics.

2 | RELATED WORK

The considerable volume of work on VNE can be categorized into optimization-based solutions[5,13,14] and graph analysis–based methods found in most heuristic strategies.[3,15,16] More details can be found in a recently published survey.[17] Optimization-based VNE proposals formulate VNE as a combinational optimization problem, then solve it exactly by integer linear programming (ILP). The optimized objects contained in most known VNE schemes involve parameters of particular interest to users and ISPs, such as maximal provider revenue, highest AR, and minimal embedding cost.[1] In addition, concerning other factors, a minimization model of energy-efficient virtual node embedding was constructed and solved to yield minimal energy dissipation.[14] Chowdhury et al[5] formulate VNE as a mixed-integer program via SN augmentation and devise two online VN embedding algorithms, deterministic D-ViNE and randomized R-ViNE, for solving the linear program with relaxed integer constraints. Jarray and Karmouch[13] decompose an overall VNE problem into a main problem with constraints on the availability of substrate resources and a pricing problem with constraints on the embedding of VNRs. In the process, they append an additional column to the constraint matrix of the primary object using column-generation technology.

Finding the optimal VNE solution via optimization methods is compute-intensive, as the problem is NP-hard. This becomes evident when we treat VNE as seeking a p-homomorphism between a weighted graph H and a subgraph of another weighted graph G. Defining p-homomorphism formally, Fan et al[18] have proven that it cannot be computed in polynomial time. Given the NP-hardness of VNE, increasing effort has been expended on designing heuristic solution algorithms.[3,15,16] Yu et al[15] aim for maximal resource utilization in the SN. Hence, the virtual nodes are greedily mapped to the substrate nodes with maximum available resources, so as to preserve resources at more resource-limited nodes. Then, each virtual link is mapped to the shortest path between two substrate nodes that embed the two endpoints of the virtual link.

The correspondence between VN and SN nodes can be established more efficiently if both can be ranked according to some topology-based measure. Cheng et al[3] consider topological properties of SN nodes as an important VNE parameter in the node-mapping stage. They rank all virtual and substrate nodes according to their relative importance. The relative importance $r(u)$ of each node u is given by $r(u) = r_0(u) + r_j(u) + r_f(u)$, where $r_0(u)$ is the product of node CPU cycle and the sum of bandwidths on outgoing links, $r_j(u)$ (jump probability) reflects the weighted importance of all reachable nodes, and $r_f(u)$ (forward probability) estimates the weighted importance of out-neighbors. Then, virtual and substrate nodes are sorted separately in nonincreasing order of node ranks. Consequently, higher-ranked virtual nodes have priority to be mapped to higher-ranked substrate nodes. Mapping of virtual links is done by shortest-path algorithm if there is no path splitting. Moreover, pertaining to ranking of nodes, Zhang and Gao[19] apply a topological potential function to calculate the topology importance of SN nodes, and then rank them, with mutual influence between a VN node and its objective nodes taken into account. Cao et al[20] consider the network topology attribute and network resource, including five important network topology attributes and global network resources, and propose a novel node ranking–based VNE algorithm called VNE-NTANRC to rank all substrate and virtual nodes before VNE process.

Another way to find the topological correspondence of nodes and links between VN and SN is through subgraph isomorphism. Lischka and Karl[16] detect subgraph isomorphism between topologies of VN and SN to discover the correspondence between nodes and links in the same stage. Evaluations show that, compared with two-stage approaches for solving large-scale VNE problems, use of subgraph isomorphism detection is more efficient.

A widely-applied method of machine learning, that is, pattern matching, has also been advocated for finding the topological correspondence of nodes and links between VN and SN, as exemplified by the proposal of Cao et al.[21] To tackle the distributed VNE problem, Beck et al[22] design a distributed and parallel framework called DPVNE to implement a VNE, in which several VNE algorithms are run to map VNs into SN in a distributed manner. The VNE strategies with time-dependent VNRs are investigated by Zhang et al.[23] This section foregoes the details of the latter approaches, given that this paper is primarily interested in VNE approaches based on node ranking.

Wang et al[24] propose a VNE algorithm marked as k-core in large-scale network. They use k-core theory, where k-core is the maximum subgraph with node connectivity at least k, to associate different VN topologies to specific SN component, and embed VNRs onto different SN components, respectively.

A main difference between algorithm called as GLNR-SP proposed in this paper and the known node ranking algorithm in the work of Cheng et al[3] is that GLNR-SP does not compute the node ranks iteratively, thus improving efficiency by avoiding the convergence of the random walk to a stationary distribution. Furthermore, GLNR-SP generates higher-quality solutions. One source of efficiency is the fact that GLNR-SP inspects the potentially large number of SN nodes just once for degree estimation; it then executes all the matrix operations on a subgraph of SN, which has the same size as the VN to be embedded, generally a much smaller network than SN.

3 | PROBLEM FORMULATION

We list the mathematical notations used in our formulation of VNE in Table 2, for ready reference.

The substrate and virtual networks can be abstracted as graphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$, respectively, where $V(\bullet)$ denotes the set of nodes and $E(\bullet)$ the set of links in the graph of interest. Let the term "constraint" describe the resource or demand of a node or link, depending on whether the described object belongs to the substrate or virtual network, and let the function $c(x)$ express the constraint at each network entity x. Then, the problem of embedding H into G can be modeled as finding a mapping $\phi : H \rightarrow G$ described as the following.

For an arbitrary entity $h \in H$, if $h \in V(H)$ then $\phi : \phi(h) \in V(G)$ defines the virtual node mapping from $V(H)$ to $V(G)$; else if $h = (u, v) \in E(H)$ and $\phi(u) = x, \phi(v) = y$ then $\phi : \phi(h) \in E(x, y)$, where $E(x, y)$ denotes the set of links on paths

TABLE 2 Mathematical notations

G	Substrate network graph
H	Virtual network graph
S	A subgraph of G to embed H
$V_H(S)$	Node subset of S for mapping nodes of H
$V_P(S)$	Node subset of S for mapping links of H
$\Delta(v)$	Set of neighbors linked to node v
$\delta(v)$	Unweighted node degree at v
$E(v)$	Set of edges incident to node v
$r(G)/r(H)$	List of (node, rank) pairs in G/H
$r(G)[u]$	Node-rank pair term indexed by u
$\lceil c(\cdot)\rceil / \lfloor c(\cdot)\rfloor$	Maximal/minimal constraint in element (node or link) "·"
$\varphi(h) \to g$	Virtual network mapping from virtual entity h to resource entity g; entity refers to node or link.
$c(v)/c(l)$	Constraints on network node v or link l
$c(g \to h)$	Resource in entity g mapped to h
$a(g)$	Resource availability in entity g of G
$p(u, v)$	Shortest path between nodes u and v
$E(u, v)$	Set of links on path $p(u, v)$
A_H/A_G	Weighted adjacency matrix of graph H or G
$L(G)$	Laplace matrix of graph G
$\mathcal{G}_\beta$	Discrete Green's function with parameter β
T	Transition possibility matrix T of a general random walk on G

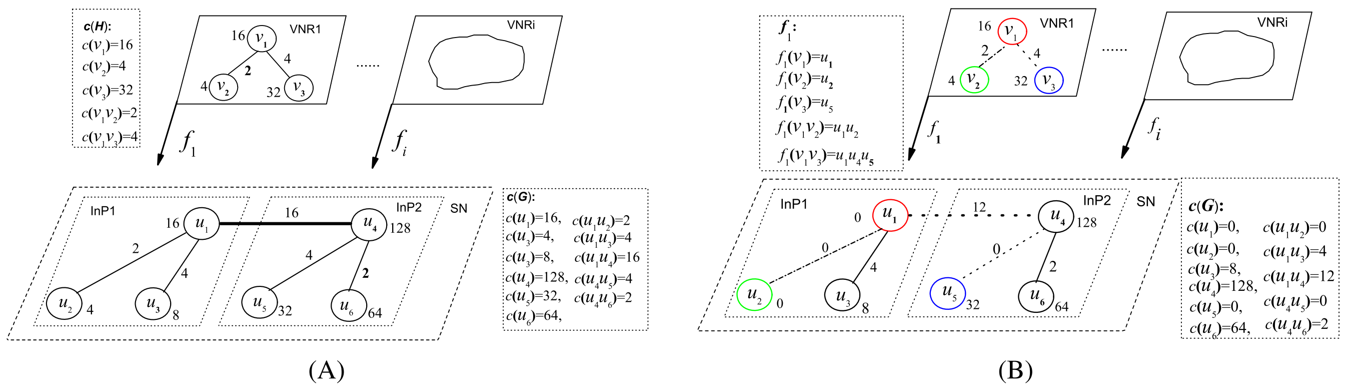

FIGURE 1 An example for our two-level architectural model for VNE, with the correspondence between virtual edge and physical edge for InP2 omitted to avoid clutter. Numbers next to circles indicate the capacity of routing/switching devices and the ones next to the links represent transmission bandwidths, in Gbps. Virtual-to-physical node correspondence is represented by distinct colors and edge correspondence is denoted by line type (dotted or dashed). A, Embedding VNR1 to SN by f_1; B, VNR1 embedded to SN by f_1

from x to y, defines the virtual link mapping from $E(H)$ to $E(G)$. The mapping ϕ from H to G must satisfy the constraint conditions: $\forall h \in H, c(h) \le c(\phi(h))$.

The "constraint conditions" are essentially that each virtual element is mapped in such a way that its demand is less than the capacity of the substrate element that it is mapped on. Furthermore, cumulative allocations of the embedding must be less than the available capacity of each substrate element.

Figure 1 offers an intuitive view of the two-level architectural model for VNE through an example. Figure 1A shows an example of a VN consisting of nodes v_1, v_2, v_3 and links v_1v_2, v_1v_3, which is to be embedded to a SN with nodes u_1, u_2, u_3 and links $u_1u_2, u_1u_3, u_1u_4, u_4u_5$, and u_4u_6. The physical node u_1 has resource 16 and the virtual node v_1 has demand 16. The substrate resource in node u_1 and the virtual demand in node v_1 can be expressed as $c(u_1) = 16$ and $c(v_1) = 16$, respectively. Likewise, the physical link u_1u_2 has bandwidth resource 2 and the virtual link v_1v_2 has bandwidth demand 2. The substrate resource on link u_1u_2 and the virtual demand on link v_1v_2 can be denoted as $c(u_1u_2) = 2$ and $c(v_1v_2) = 2$, respectively.

4 | SCHEME PRESENTATION

4.1 | Preliminary

To formulate the VNE process, the weighted adjacency matrix of graph G is denoted with A_G, representing the weights distribution of entities on digraph G. Generally, A_G can model the communication topology of digraph G with more details than the commonly used standard adjacency matrix. With $a(x, y)$ denoting the residual available bandwidth on link (x, y), that is, the capacity constraint of link (x, y) minus the already allocated resource quantity, the (x, y)-entry of matrix A_G is defined as

$$A_G(x, y) = \begin{cases} a(x, y), & \text{if } x \sim y; \\ 0, & \text{else.} \end{cases}$$

For virtual network H, A_H is still defined as above, but $a(x, y) = c(x, y)$ represents the quantity of resource demand on (x, y), regardless of prior allocation.

4.2 | The main algorithm

The main algorithm GLNR-SP, reflecting our strategy for embedding virtual networks onto a massive SN, includes two procedures: GenSubset and LocalRank. The GenSubset procedure finds a subset $V(S)$ of substrate nodes spanning a subgraph S of the SN G. Generally, S contains candidates of all virtual nodes, and S has a far smaller scale than $V(G)$ in case of massive G (see procedure GenSubset for details of generating $V(S)$). Therefore, S can provide enough available node and link resources to accommodate H, and embedding H to G can be confined to embedding H to S. This can lead to significant complexity reduction when comparing algorithm GLNR-SP with other existing algorithms. Subsequently, procedure LocalRank in algorithm GLNR-SP is called for ranking all virtual nodes $V(H)$ and substrate candidates $V(S)$ selected out. In this stage, local ranking (see[9,25,26]) is performed to rank the nodes of both S and H. Then, all ranked nodes in $V(H)$ are mapped to the corresponding ranked nodes in $V(S)$ according to the associated weights generated by procedure LocalRank (see procedure LocalRank for details of node ranking). In what follows, we provide details for procedures GenSubset and LocalRank after describing the process of node mapping when procedures GenSubset and LocalRank are successfully implemented. Suppose that the process of node ranking on S and H outputs the ranking vectors $r(S)$ and $r(H)$, and the elements of $r(S)$ and $r(H)$ are pairs of the form (node, rank), that is,

$$r(S) = [(s_1, r(s_1)), (s_2, r(s_2)), \ldots, (s_{|S|}, r(s_{|S|}))], s_i \in V(S);$$

then, the rank of a node x can be retrieved by $r(x)$.

Algorithm GLNR-SP maps the virtual node $v \in V(H)$ to the candidate substrate node $s \in V(S)$ according to their ranked values in $r(H)$ and $r(S)$, as long as the resource of s satisfies the demand of v. Node mapping $\phi : V(H) \rightarrow V(S)$ can thus be formalized as the following.

$$\text{For } (v \in V^{\uparrow}(H)), s \in V^{\uparrow}(S)), \text{ if } c(v) \leq (s) \text{ then } \phi(v) = s.$$

Once node mapping is completed, the variable *nodemapping* stores the node pair list of virtual and mapped substrate nodes. Procedure kShortestPathLinkMapping (nodemapping, H, k) maps each virtual link (u, v) onto the shortest substrate path $p(u, v)$ with length k. Details of algorithm steps are provided in the form of pseudocode for GLNR-SP. In particular, please note the important line 10: delete$(u, r(u))$ of the algorithm. Assuming that some reasonable subgraph has been computed, without line 10: delete$(u, r(u))$, the main algorithm GLNR-SP will nearly always yield infeasible embeddings, because for each virtual node, the list of substrate nodes is iterated according to the rank and if the capacity is sufficient, the virtual node is mapped onto the respective substrate node. However, as nowhere the capacities are reduced, and, by design, GenSubset only returns substrate nodes capable of hosting any virtual node, all virtual nodes will be mapped onto a single substrate node. This will lead to exceeding the capacities in any reasonable scenario. This case could be easily identified a priori by just checking if a substrate node may host all virtual nodes.

```
Algorithm GLNR-SP:
   Input: virtual network $H$=($V(H)$, $E(H)$) and substrate network $G = (V(G), E(G))$
   Output: virtual network mapping $\phi$, which embeds $V(H)$and $E(H)$ onto $V(G)$ and $E(G)$, respectively
1  $S \leftarrow$ GenSubset($H, G$);   /* Generate the $H$-Admittable Embedding Subgraph $S$ */;
2  $r(S) \leftarrow$ LocalRank($S$);   /* Local ranking on graph $S$ */;
3  $r(H) \leftarrow$ LocalRank($H$);   /* Local ranking on graph $H$ */;
4  $V^{\uparrow}(S) \leftarrow$ sort($V(S)$) by $r(S)$;  /* Sorting nodes in $V^{\uparrow}(S)$ by values of local ranking on $S$ */;
5  $V^{\uparrow}(H) \leftarrow$ sort($V(H)$) by $r(H)$;  /* Sorting nodes in $V^{\uparrow}(H)$ by values of ranking on $H$ */;
6  for each ($v$ : $V^{\uparrow}(H)$) do
7      for each ($u$ : $V^{\uparrow}(S)$) do
8          if ($c(u) \geq c(v)$) then
9              $\phi(v) \rightarrow u$;   /* If $u$ is available to $v$, mapping $v$ to $u$ */;
10             delete($u, r(u)$) in $r(S)$;   /* $u$ will not be considered as candidate of next VN node */;
11             nodemapping.add($v, u$);   /* add the node pair $(v, u)$ to the mapped node list nodemapping */;
12         end
13     end
14 end
15 kShortestPathLinkMapping(nodemapping, $H$, $k$);   /* execute link mapping */;
```

4.3 | Generating the H-admittable embedding subgraph of G

Identifying an AES can essentially be resolved into finding a substructure of the substrate graph as close as possible to a clique of size "number of virtual nodes." Network elements of such a substructure can be selected by pruning network elements not supporting the maximal capacity. This is important as finding a clique itself is NP-hard and the process of "estimating" a substructure close to it should be investigated. Procedure GenSubset attempts to find a subset of substrate nodes that span a subgraph S of substrate G such that S is smaller than G, but contains candidates of all virtual nodes. Let $p(x, y)$ be the shortest path between nodes x and y, and $E(x, y)$ denote the set of links on $p(x, y)$. To pick nodes from G to generate S such that H can be embedded into S with a high probability of success globally, the expected node set $V(S)$ should contain at least as many as $|H|$ nodes. In addition, nodes and links in S must have sufficient residual resources to fulfill the corresponding demands from H. In other words, for any VN elements (nodes and links) $h \in H$, $c(h) \leq c(\phi(h))$ must hold. In particular, when $\phi(h)$ represents a substrate path, the virtual link h cannot be successfully embedded unless each link of this path has enough resources available. Thus, it is needed that

$$c((u, v)) \leq \min\{a(l) | l \in E(\phi(u), \phi(v))\},$$

where $a(\bullet)$ and $c(\bullet)$ represent the available resource and original constraints (resource capacity for substrate entities and resource demand for virtual ones) in network entity "$\bullet$". Besides, S must be connected to ensure that there is at least one path for successful link mapping between arbitrary pairs of nodes. Moreover, a highly connected S leads to short paths for node pairs.

A subgraph S of G with the features discussed above is called as an H-AES of G. For generation of such a subgraph, procedure GenSubset selects a subset $V(S)$ of $V(G)$ as the vertex set of the expected graph in advance. Procedure GenSubset sets $V(S)$ as null initially, then gradually adds qualified substrate nodes to $V(S)$. For adding substrate nodes with sufficiently available resources to $V(S)$, the classical topological sorting algorithm is applied to G to generate the sorted $V(G)$ nodes list denoted as $V^{\uparrow}(G) = \{u_0, u_1, \ldots, u_{n-1}\}$. To make topological sorting feasible, the substrate graph G should be converted into a directed graph $\vec{G}$ by changing all undirected edges (x, y) to the directed edges $\overrightarrow{(x, y)}$ if $a(y) \geq a(x)$, and $\overrightarrow{(y, x)}$ otherwise. Topological sorting of $V(G)$ into $V^{\uparrow}(G)$ places substrate nodes $V(G)$ into nondescending order in term of available resources and topological structure. Then, GenSubset picks nodes from $V^{\uparrow}(G)$ and links from $E(G)$ to form S. To accomplish this, GenSubset traverses all virtual nodes and links, finding the maximal node and bandwidth demands $\lceil c(h) \rceil$ and finding the minimal node and bandwidth demands $\lfloor c(h) \rfloor$. Then, it initializes candidate node set $V(S)$ to enclose the first substrate node with quantity of available resource greater than the minimal node demand $\lfloor c(h) \rfloor$, that is,

$$S = \{u_k | a(u_k) \geq \lfloor c(h) \rfloor, a(u_{k-1}) \leq \lfloor c(h) \rfloor \}.$$

Then, procedure GenSubset consecutively selects substrate nodes fulfilling the demand from $V^{\uparrow}(G)$ for S until $V(S)$ includes all candidates of $V(H)$. For ensuring that nodes and links in S have sufficient residual resource to fulfill the corresponding demands from H, candidates added to $V(S)$ are inspected in order in $V^{\uparrow}(G)$ to ensure that they are capable of fulfilling the maximal demands in node capacity or link bandwidth. The process is formally describe as

$$\text{if } (c(v) \leq c(u)) \&\& (\delta(v) \leq \delta(u)) \&\& (\max\{c(l_h) | l_h \in E(v)\} \leq \min\{c(l_g) | l_g \in E(g)\}),$$

$$V(S) = V(S) \cup \{u\}.$$

To ensure connectivity and short paths between node pairs of S, when selecting a new node u from $V^{\uparrow}(G)$ to be added to $V(S)$, a node u directly linked to at least one node s of $V(S)$ will be chosen first, if node u and link (u, s) provide sufficient resource. This can be formulated as

$$u \sim S \&\& \ \max\{c(l_h) | l_h \in E(v)\} \leq \min\{c(E(u, S))\}, \text{ then } V(S) = V(S) \cup \{u\} \text{ and } E(S) = E(S) \cup \{(u, s) | s \in S\}.$$

If no edges linking node u and nodes of $V(S)$ exist, GenSubset selects the one node of $V^{\uparrow}(G)$, which has the shortest average path length to nodes of $V(S)$. If the candidate selected for inclusion in $V(S)$ is not linked directly to any node in $V(S)$, but is connected to some node via a path, GenSubset must also add all nodes on this path to $V(S)$ to guarantee the connectivity of S. For distinguishing these added path nodes from those that are directly linked, GenSubset separates the added nodes on path from $V(S)$, and denotes them with $V_P(S)$. Denoting the substrate nodes added to $V(S)$ as embedding candidates rather than path transition as $V_H(S)$, it holds that $V_H(S) = V(S) - V_P(S)$.

If nodes x and y have identical available resource: $a(x) = a(y)$ as well as the same shortest average path length, how does GenSubset select the next node? The strategy of GenSubset prefers to the node with higher connectivity, thus if node x has more adjacent nodes in $V(S)$, then x will be preferred for inclusion in $V(S)$ before y. Ultimately, the procedure for selecting $V(S)$ terminates once $V(S)$ contains all candidates of $V(H)$. Algorithmically, the expected set of nodes $V(S)$ can be obtained by implementing steps in corresponding procedure **GenSubset**, listed as executable pseudocode in GenSubset.

In way of elaboration, GenSubset does not have a simpler process for constructing S as the induced subgraph of $V(S)$in G because such a simply-produced S may lead to no AES in G. An example follows. The SN: $V(G) = u_1, u_2, u_3$, $E(G) = (u_1, u_2), (u_2, u_3)$, $a(u_1) = 30$, $a(u_2) = 10$, $a(u_3) = 20$, $a(u_1, u_2) = 20$, $a(u_2, u_3) = 20$. Virtual network: $V(H) = u_1, u_2$, $E(H) = u_1, u_2$, $c(u_1) = 15$, $c(u_2) = 15$, $c(u_1, u_2) = 10$. The substrate node u_2 does not have enough capacity, but the set $\{u_1, u_3\}$ is disconnected. Note that, $v_1 \to u_1$, $v_2 \to u_3$, $(v_1, v_2) \to (u_1, u_2), (u_2, u_3)$ is a feasible embedding.

```
Function Digraph($G$)
 Input: $G = (V(G), E(G))$
 Output: The directed graph $\overrightarrow{G}$
1 $\overrightarrow{G}$=($V(G)$,null);
2 for each ($(x, y)$ : $E(G)$) do
3     if ($a(x) \leq a(x)$) then
4         add directed edge $\overrightarrow{(x, y)}$ to $E(\overrightarrow{G})$;
5     end
6     else
7         add directed edge $\overrightarrow{(y, x)}$ to $E(\overrightarrow{G})$;
8     end
9 end
10 return $\overrightarrow{G} = (V(G), E(\overrightarrow{G}))$;
```

```
Procedure GenSubset($H$, $G$)
  Input: virtual network $H$=($V(H)$, $E(H)$) and substrate network $G$= ($V(G)$, $E(G)$);
  Output: $S$: the $H$-AES of $G$;
1 $\overrightarrow{G}$=Digraph($G$);
2 $\overrightarrow{H}$=Digraph($H$);
3 $V^{\uparrow}(G)$=TopologicalSorting($\overrightarrow{G}$);
4 $V^{\uparrow}(H)$=TopologicalSorting($\overrightarrow{H}$);
5 for each ($v$ : $V^{\uparrow}(H)$) do
6     for each ($u$ : $V^{\uparrow}(G)$) do
7         if $(c(v) \leq c(u))$ &&$(\delta(v) \leq \delta(u))$ && $(\max\{c(l_h)|l_h \in E(v)\} \leq \min\{c(l_g)|l_g \in E(g)\})$ then
8             if ($V(S)$= null) then
9                 $V(S) = V(S) \cup \{u\}$;
10            end
11            else if ($u \sim S$ && $\max\{c(l_h)|l_h \in E(v)\} \leq \min\{c(l_g)|l_g \in E(u, S)\}$ then
12                $V(S) = V(S) \cup \{u\}$;
13                $E(S) = E(S) \cup \{(u, s)|s \in S\}$;
14                break;
15            end
16        end
17    end
18 end
19 return $S$;
```

4.4 | Local random walk on the *H*-admittable embedding subgraph of *G*

Once procedure GenSubset completes successfully, the subset $V(S)$ has been selected from $V(G)$, the H-AES of G can be assigned as the subgraph $S = (V(S), E(S))$. To discover a near optimal correspondence between S and H for node mapping, a natural inclination is to consider this problem as weighed graph matching. Unfortunately, embedding H to S has differences with graph matching between two graphs having equal number of nodes, given that a virtual link (u, v) can be embedded onto a substrate path (x, y). Hence, procedure LocalRank associates nodes and links of H with nodes and links of S, according to the corresponding weight values in the local node raking vectors $r(H)$ and $r(S)$. To locally rank all substrate nodes $V(S)$, LocalRank arranges them in nonincreasing order according to the local node ranking vector $r(S)$, which reflects resource availability. The node ranking vectors $r(H)$ and $r(S)$ are solutions of the stationary distribution of random walks on H and S, in which entry $r(x)$ estimates the potential energy x relative to other nodes. Generally, the VN has a far smaller scale than that of the SN, which prompts us to calculate $r(S)$ more precisely, that is, by employing more elaborate topological information of H and S to perform the procedure of local random walk. Next, the general notion of random walk on graph G will be defined, and then be specialized to S. The node ranking vector $r(H)$ is obtained from a general random walk on graph H and $r(S)$ from a local random walk on subgraph S. The transition possibility matrix T of a general random walk on G is defined[9] as

$$T = D^{-1}A(G), \tag{1}$$

where D denotes the diagonal matrix with entries $D(x, x) = d(x)$, $d(x)$ being the weighted degree of vertex x defined as

$$d(x) = \sum_{\substack{y \sim x \\ y \in V(G)}} a(y). \tag{2}$$

A lazy random walk M on G can be defined[9,25,26] by $M = (I + T)/2$, where I is the identity matrix. Then, utilize personalized PageRank, generalized to measure the node ranking value $r(x)$ of x, determined from equation

$$r = \alpha s + (1 - \alpha)rM, \tag{3}$$

where parameter $\alpha \geq 0$ is the *jumping constant* that scales the rate of propagation and b expresses a user-defined initial probability distribution vector. A vector is understood to be in row form throughout this paper, unless otherwise specified. The definition above leads to an equivalent description of the recurrence form for r

$$r = \alpha \sum_{k=0}^{\infty} (1-\alpha)^k M^k. \tag{4}$$

To obtain r from the equation above, resort to the discrete Green's function $\mathcal{G}$, which operates as the inverse of the Laplacian operator on the space orthogonal to the null space of the Laplacian (see the works of Chung et al[10-12] for more details of discrete Green's function). Let β be the constant $\beta = 2\alpha/(1-\alpha)$ and $L(G)$ be the Laplacian matrix of graph G. The discrete Green's function, generalized as $\mathcal{G}_\beta$ by Chung,[10-12] is the inverse of $\beta I + L$, expressed as

$$L = I - T \quad \text{and} \quad \mathcal{G}_\beta = (\beta I + L)^{-1}. \tag{5}$$

Thereby, the rank vector r defined in Equation (3) can be solved using discrete Green's function $\mathcal{G}_\beta$ through the equation

$$r = \beta s \mathcal{G}_\beta. \tag{6}$$

Unfortunately, obtaining r from Equation (6) generally entails $O(n^3)$ time overhead to calculate the inverse of $\beta I + L$, namely, $\mathcal{G}_\beta$. Such a time complexity is usually unacceptable in fairly wide range of application domains. Thus, a directed graph $\overrightarrow{G}$ can be relaxed to an undirected version to rank the nodes globally, and still treat it as a directed graph in stage of node ranking on subgraph S. Such a relaxation requires us to rewrite the entity $a(x,y)$ of $A(G)$ as

$$a(x,y) = (a(x,y) + a(y,x))/2.$$

Assume that graphs G and H are both strongly connected and that the random walks on G and H are both aperiodic, which means that the matrix T has only one eigenvalue with absolute value 1. Then, the stationary distribution r of random walk is the Perron vector of T, namely, the eigenvector of T corresponding to the maximal eigenvalue 1. In this case, there is a closed form solution for r, given by

$$r(x) = d(x) / \sum_{y \in V(G)} d(y), \tag{7}$$

where $d(x)$ is defined by Equation (2).

To commence the local ranking on S, let $\partial S = \{v | v \notin S \wedge v \sim u \in S\}$ express the set of nodes located at boundary of S. Then, initialize the vector b, also viewed as a function defined on $b : S \cup \partial S \rightarrow R$, to satisfy the Dirichlet boundary condition,[9,25] that is,

$$x \in \partial S, b(x) = 0.$$

When defined as follows, b will meet the Dirichlet boundary condition:

$$\text{If } x \in S \text{ then } s(x) = d(x)/\sigma(S), \text{ else } s(x) = 0, \text{ where } \sigma(S) = \textstyle\sum_{x \in S} d(x).$$

A function f satisfying the Dirichlet boundary condition on S ensures that the relevant operators, such as the transition probability and Laplace matrices, can be represented by their submatrices restricted to vertices of S when these operators act on f. Such a function would simplify node ranking on graphs. For locally ranking all nodes in $V(S)$, initially, generate the submatrix $A(S)$ by extracting the rows and columns corresponding to the vertices in S from the adjacency matrix A. Borrowing the notation of Matlab, this means $A(S) = A(S, S)$. Use the same approach to define the diagonal matrix $D(S)$ with diagonal entries being vertex degrees, namely,

$$D(S) = \text{diag}\{d(x) | x \in S\}.$$

The local transition probability matrix $T(S)$, the local Laplacian $L(S)$, and the local discrete Green's function $\mathcal{G}(S)$ can be produced in a similar manner,[9,25,26] by using equation

$$T(S) = D(S)^{-1}A(S).$$

Eventually, local node ranking vector $r(S)$ can be derived from the equation

$$r(S) = \beta s\mathcal{G}(S).$$

Attention to two points with respect to local random walk is necessary. To avoid global operations as much as possible, one can generate $A(S)$ directly from the subgraph S, which has nodes in $V(S)$ and edges of G with both endpoints in S, rather than extracting it from $A(G)$. Moreover, compute $d(x)$ for $x \in S$, but remember that the node degree of $x \in S$ should be computed by

$$d(x) = \sum_{y\sim x, y\in G} a(y).$$

In other words, summing over all nodes in G, not only over ones in S, that implies that even if nodes outside of S have been ignored, they still affect the local random walk on S because the original edges including in $E(S, \partial S)$ still contribute to $r(S)$. The executable pseudocode of procedure LocalRank is as follows.

```
Procedure LocalRank($S$)
  Input: $S$: the $H$-AES of $G$;
  Output: the node ranking vector $r(S)$;
1. $T(S) = D(S)^{-1}A(S)$;
2. $L(S) = I(S) - T(S)$;
3. $\mathcal{G}(S) = (\beta I(S) + L(S))^{-1}$;
4. for each ($x$ : $S$) do
5.     $s(x) = d(x)/\sigma(S)$;
6. end
7. $r(S) = \beta s\mathcal{G}(S)$;
8. return $r(S)$;
```

5 | EVALUATION

Results of experiments discussed in this section demonstrate efficiency and quality of our scheme in terms of multiple representative evaluational metrics such as runtime, acceptance ratio (AR), cost/revenue ratio (C/R), and node utilization ratio (NUR). All simulations are conducted under the platform consisting of software Eclipse Oxygen IDE running in 64-bit Windows 7 over hardware CPU Intel(R) Core(TM) i5 4590 @3.3 GHz with 16.0GB RAM. In addition, simulations generate, analyze, and compare the results relying on recently updated simulation software package Alevin 2.2, developed by Beck et al,[27] recognized as a significant simulation framework for evaluating VNE algorithms.

Algorithm GLNR-SP proposed in this paper is realized with Java language within multiple experimental settings. Experimental results evaluate the performance and quality of GLNR-SP in terms of multiple evaluational metrics. The experimentation process can be divided into scenario generation, algorithm setting and execution, and algorithm evaluation, with various experimental configurations. The experimental configurations are further detailed in Table 3. Subsequently, algorithm GLNR-SP is compared with the representative VNE algorithms: DViNE-SP, GAR-SP, RW-MM-SP and vnmFlib, that have been cited as the focus of considerable VNE research, and two recently suggested VNE algorithms: VNE-NTANRC and k-core, in terms of runtime, VNR acceptance ratio, C/R, and NUR, factors recognized as effective means of assessing VNE algorithms. Finally, the results of comparison have been plotted to reveal the effect of varied scales of VN and SN on VNE performance and quality.

Scale	SN size	VN size	VNs num.	Demand	Resource
Small-medium	$5k$	$2k$	5	[1-10]	[1-100]
Massive	1000	$2k$	5	[1-10]	[1-100]

TABLE 3 Experimental parameters of network generation, k indicating the experimental period

TABLE 4 Experimental algorithms for evaluation

Algorithm	Reference	Brief description
DViNE-SP	Chowdhury et al[5]	VNE with coordinated strategy in two stages where node mapping is implemented by mixed integer programming (MIP) and link mapping with k-shortest paths
GAR-SP	Yu et al[15]	VNE preferentially using available resources for node mapping and k-shortest paths for link mapping
RW-MM-SP	Cheng et al[3]	VNE ranking nodes with topology properties for node mapping and k-shortest paths for link mapping
vnmFlib	Lischka and Karl[16]	VNE based on subgraph isomorphism detection with nodes and links mapping in the same stage
VNE-NTANRC	Cao et al[20]	VNE concerned on network topology attribute and network resource
k-core	Wang et al[24]	VNE for large-scale network using k-core theory

5.1 | Scenario generation

When setting up an experimental scenario, network topology consists of randomly generated VN topologies and artificially configured SN topologies Hub-Star. Random VN topologies are generated with a probability 0.5 of connecting a pair of nodes. Note that resulting VN could possibly be disconnected. In such a case, network generator inspects the isolated vertices and randomly add edges until a connected VN is obtained. For generating Hub-Star SN, network generator first randomly produces a k-vertex graph whose vertices model hub nodes arising in real ISP network, following each step of k steps, randomly add $r_i(0 \leq i \leq k)$ of vertices to graph, and connect newly added r_i vertices to one hub node h_i to form a h_i-centralized star subgraph. Consequently, the resulting Hub-Star SN topology is characterized by a number of hubs, a high number of nodes is connected to one or more hubs, which have low internodes distance and heavy traffic on hubs. This approach of SN generation captures the significant topological characteristics of the Star and the hub & spoke networks,[4] and it relatively reliably imitates the most traditional ISP network topologies arising in real communication. After completion of network topology, CPU and bandwidth resources of SN are generated randomly as real numbers in the interval $[1, 100]$; and CPU and bandwidth demands of VN are generated randomly as real numbers in the interval $[1, 10]$. The simulation proceeds with varying the network scale with experimental times. Table 3 lists more details of network generation.

5.2 | Algorithm configuration

Four representative VNE algorithms (DViNE-SP, GAR-SP, RW-MM-SP, and vnmFlib) and two recently suggested VNE algorithms (VNE-NTANRC and k-core) are compared with algorithm GLNR-SP for evaluation of performance and quality, as proposed in other works[3,5,15,16,20,24] and briefly described in Table 4. Consideration for selection of algorithms involves mainly two factors. First, they have received considerable attention through respectful citations by Google Scholar.* Second, they can be realized completely on the software package Alevin 2.2 such that algorithms can be compared in a fairly impartial environment. All algorithms used for experiments were executed on the same scenarios and parameter configurations. All algorithms chosen for evaluation were run 25 times under identical scenario configurations, while increasing the sizes of SN and VN. The parameters of CPU node weights and distance were assigned as 1 and 20, respectively, without consideration of node overload.

5.3 | Evaluation results

Experiments have been conducted on small-medium–sized and massive SNs, respectively, with VNE algorithms of state of art, listed in Table 4. Originating from presentation of the experimental purpose and hypothesis, experimental results generated by algorithm GLNR-SP are compared with ones derived from other VNE algorithms with respect to embedding efficiency, quality, and scalability. Moreover, the reason behind such result has been disappeared, which is the main

*https://scholar.google.com

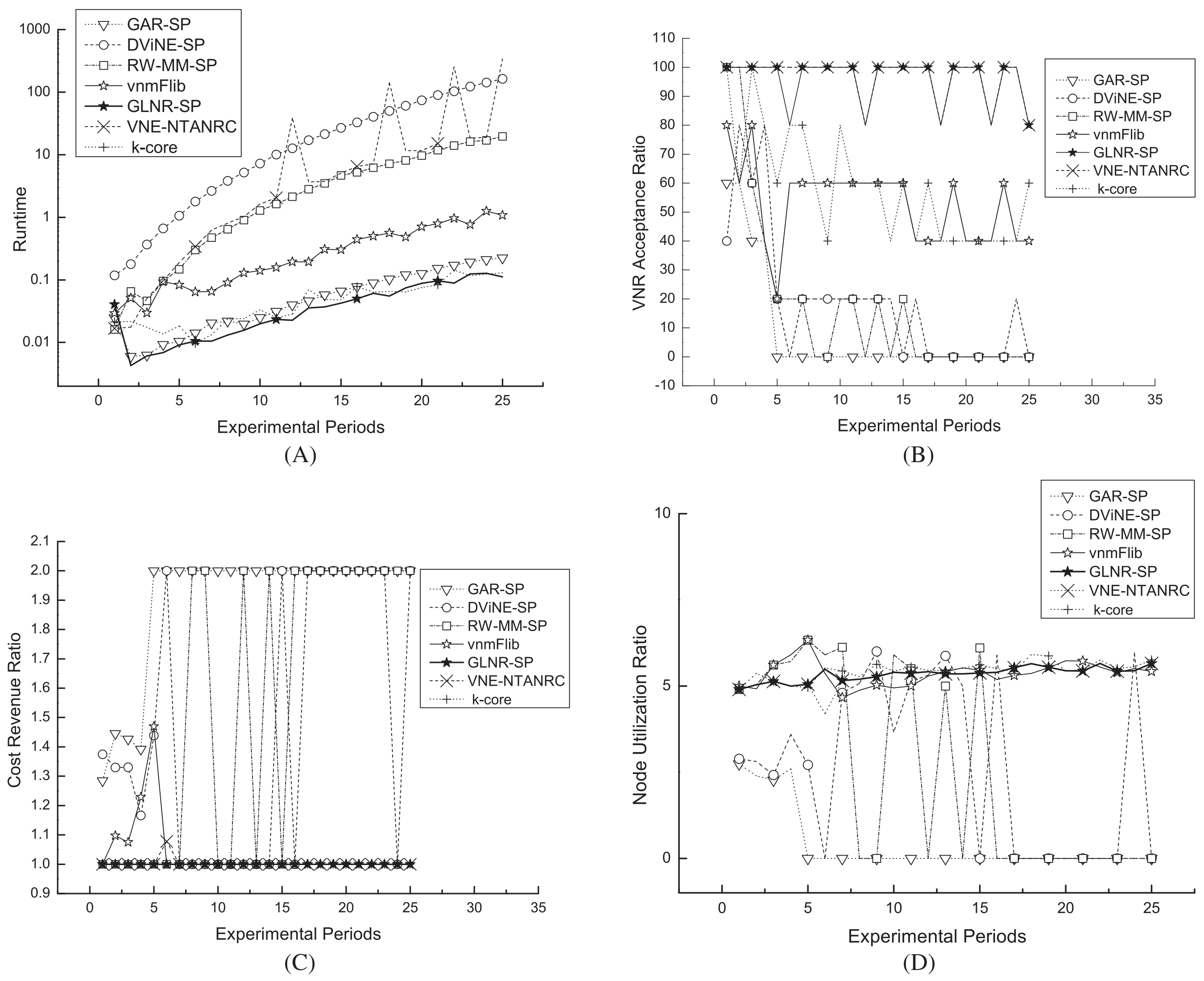

FIGURE 2 Performance and quality comparisons of GLNR-SP with representative VNE schemes for embedding $2k$-sized VN into a small-medium–sized SN with $5k$ nodes, $1 \leq k \leq 25$ indicating the experiment's range. A, Runtime; B, VNR acceptance ratio; C, Cost/revenue ratio; D, Node utilization ratio

contribution of this work. The results of comparisons with other algorithms in multiple metrics listed above are depicted in Tables 5 and 6 and in Figures 2 and 3. Tables 5 and 6 illustrate the average values of concerned metrics resulting from 25 rounds of executing the competing algorithms in cases of meso-micro scale SN and massive SN, respectively. Figures 2 and 3 offer the details for the entire period corresponding to Tables 5 and 6, respectively. Note that the invalid results due to a divide-by-zero exception have not appeared in Tables 5 and 6. Experiments consider algorithm evaluations in three dimensions: embedding efficiency, embedding quality, and scalability for addressing evaluation results.

5.3.1 | Experimental purpose and hypothesis

The process of evaluation focuses on a collection of well-recognized VNE metrics (runtime, VNR acceptance ratio, C/R, and NUR) for assessing the competitiveness of our proposal. On purpose, algorithm GLNR-SP is expected to perform better than competing ones with respect to runtime and AR, given our particular interests in trade-off of embedding efficiency and quality. Simultaneously, experiments also attempt to prevent degradation in metrics of C/R and NUR. The experimental results confirm that GLNR-SP yields the expected results in runtime and AR metrics, without a cost in quality degradation in terms of other evaluation metrics. The experimental hypotheses are listed as following H1, H2, and H3.

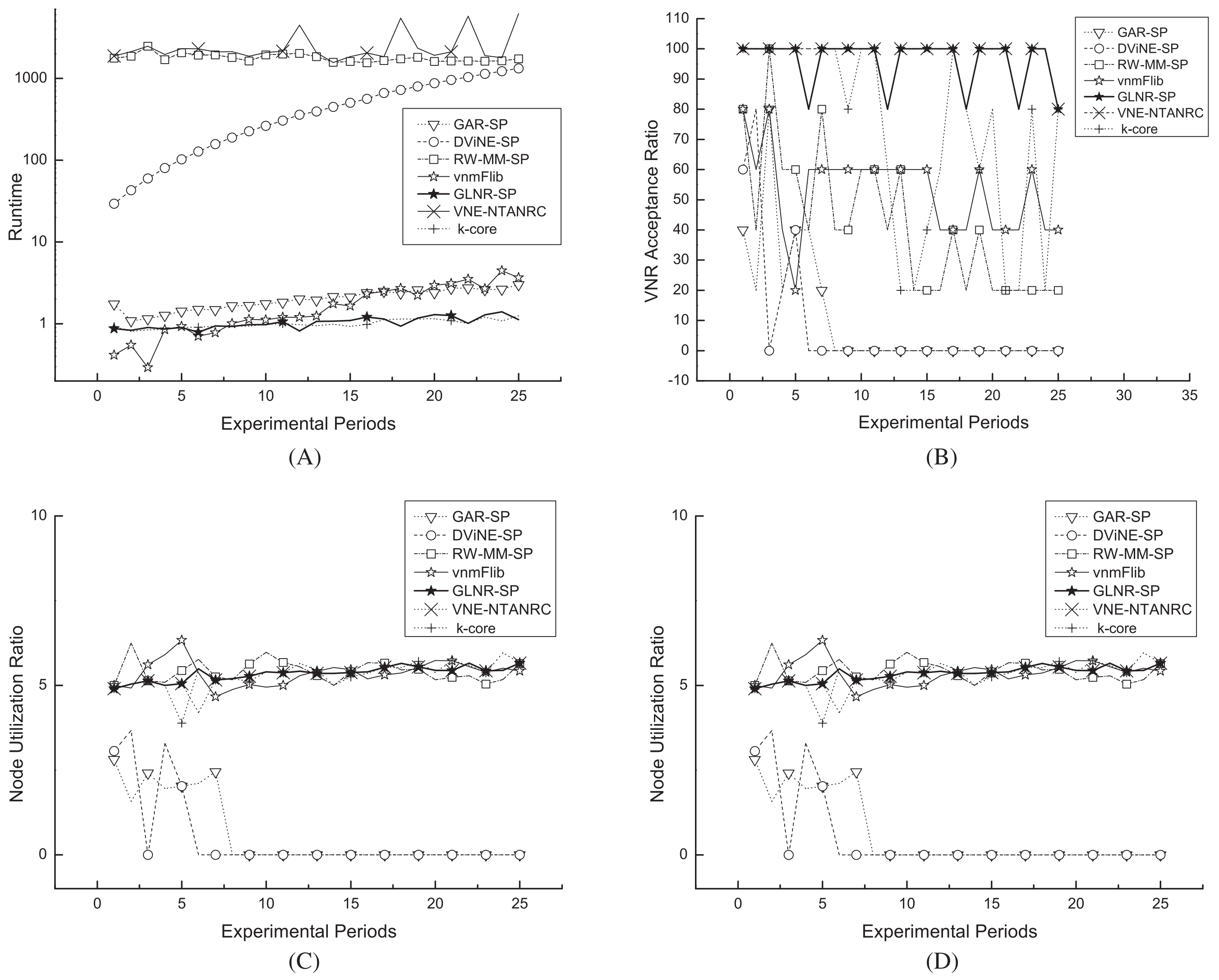

FIGURE 3 Performance and quality comparisons of GLNR-SP with five representative VNE schemes for embedding into a massive SN with 1000 nodes. A, Runtime; B, VNR acceptance ratio; C, Cost/revenue ratio; D, Node utilization ratio

Algorithm	Runtime(s)	Acceptance	C/R	NUR
GAR-SP	0.07	8.00	1.90	0.40
DViNE-SP	39.57	19.20	1.47	2.69
RW-MM-SP	5.34	17.60	1.56	2.49
vnmFlib	0.38	52.80	1.03	5.36
VNE-NTANRC	35.41	96.80	1.00	5.30
k-core	0.05	58.4	1.00	5.46
GLNR-SP	0.05	96.00	1.00	5.35

TABLE 5 Comparison of the average values of the evaluation metrics from test iterations executed on small-medium–sized SN

H1: Experiments are particularly focused on shortest-path class VNE approaches, and algorithmic versions with path splitting have not yet been considered. Furthermore, the link mapping parameter k of mapping a VN link to a length-k shortest path was set to $k = 2$.

H2: The numbers n and m of SN and VN nodes increase linearly with the iteration i of experiment according to $n = a \times i$ and $m = b \times i$, where a and b are constant factors controlling the growth of SN and VN, respectively. For configurations here, they hold values $a = 5$ and $b = 2$ and $1 \le i \le 25$) (see Table 3).

H3: The weights on node resource and link bandwidth conform to the uniform distribution in the interval [min, max].

TABLE 6 Comparison of the average values of the evaluation metrics from test iterations executed on massive SN (the SN networks are enlarged to the size of 1000), to demonstrate the scalability of our strategy

Algorithm	Runtime	Acceptance	C/R	NUR
GAR-SP	2.03	10.40	1.86	0.61
DViNE-SP	502.94	8.00	1.87	0.48
RW-MM-SP	1788.84	41.60	1.00	5.42
vnmFlib	1.80	52.80	1.03	5.36
VNE-NTANRC	2585.31	96.80	1.00	5.30
k-core	1.05	72	1.00	5.31
GLNR-SP	1.04	96.00	1.00	5.35

5.3.2 | Embedding efficiency

Experiments were planned to enlarge the scales of SN from 6 to 1000 to demonstrate the performance strength of proposed algorithm when the size of SN becomes massive. The largest size of VN is 50, emerging in the 25th iteration. The effort of testing our algorithm on larger VN was hindered by the excessive running time of the optimization-based scheme DViNE-SP. When embedding randomly generated VN with 60 nodes to the massive SN in the 30th iteration, algorithm DViNE-SP encounters efficiency problems, so the optimization package GLPK 4.7 terminates the searching for solutions due to timeout, as shown in Figure 2. However, even if algorithms are implemented on VN with scale from 2 to 50, there are apparent performance differences observed in Tables 5 and 6 and in Figures 2A and 3A. The scheme GLNR-SP exhibits an apparent improvement over the other algorithms (GAR-SP, DViNE-SP, RW-MM-SP, vnmFlib, VNE-NTANRC, and k-core) in average values with respect to all metrics (Table 5), and the maximal value regarding runtime, VNR acceptance ratio and C/R, except for the maximal NUR RW-MM-SP (Table 6). The algorithm k-core nearly approaches GLNR-SP in runtime, but it apparently degrades in AR, as seen in later section.

Algorithm GLNR-SP also demonstrates its overall advantage in process of 25 times embedding (see Figures 2A and 3A). On randomly generated VN and Hub-Star SN, through a 25-iteration test, GLNR-SP embeds 96% of VNRs within nearly average runtime 0.05 seconds, which is less than the best one GAR-SP's 0.07 seconds of other algorithms besides k-core. K-core embeds 58.4% of VNRs in nearly same runtime 0.05 seconds. The strength in runtime of our algorithm increases alone with enlargement of network scale, and our method manifests its prominence in efficiency particularly in massive SN, as listed in Table 6, where GLNR-SP embedding the maximal VN with 50 nodes into maximal SN with 1000 within 1.04 seconds. This value is clearly less than GAR-SP's 2.03 seconds, vnmFlib's 1.80 seconds, and k-core's 1.26 seconds. The other tested algorithms DViNE-SP, RW-MM-SP, and VNE-NTANRC have the runtimes of DViNE-SP's 502.94 seconds, RW-MM-SP's 1788.84 seconds, and VNE-NTANRC's 2585.31 seconds exhibited poor efficiency, which implies that they would have an unacceptable computational overhead when the scales of both SN and VN become large.

5.3.3 | Embedding quality

The AR, defined as the ratio between the number of accepted VNRs and the total number of VNRs, reflects the success rate in embedding of virtual networks. The C/R, defined as the sum of the substrate resources allocated to the VNR, represents the amount of resources used by an embedding. The revenue sums the revenue of the VNRs that were successfully mapped and the revenue of those that were not mapped. The C/R measures the proportion of cost spent in the SN, taking into account the revenue that has been mapped, the lower the C/R, the better the mapping quality. The node utilization of SN is calculated as the ratio of used CPU cycles to the number of nodes in it. It reflects the proportion of resources being utilized to meet the currently accepted VNRs. The GLNR-SP generates an average VNR acceptance ratio 96%, average C/R around 1.00, and the NUR around 5.35, higher than those of the other algorithms in small and medium-sized SN, as seen in Table 5. The GLNR-SP sustains these evaluation metrics relatively stable when the size of SN achieve to 1000 as seen in Table 6.

Figures 2 and 3 demonstrate more details that GLNR-SP produces more competitive values with regard to five embedding quality metrics than the other approaches. The average value 96.0% and maximal value 100% in AR, the average and maximal value 1.0% in C/R, and the average value 5.35% in node utilization demonstrate the best quality score in all tested algorithms, except that GLNR-SP has the average value 5.35% in node utilization lower than both RW-MM-SP's 5.42 and vnmFlib's 5.36 in a test suite of 25. Overall, evaluation demonstrates that GLNR-SP is superior to other approaches with regard to qualitative metrics.

5.3.4 | Scalability

For evaluating VNE algorithms in massive networks, the sizes of SN and VN vary in an extensive range from 2 to 50 for VN and 5 to 125 for small-medium–sized SN, 1000 for massive SN. Further enlargement of networks has been attempted, as mentioned before. When the scale of VN is set at 60 (arising in 30th iteration), the algorithm DViNE-SP terminates to search for the optimal solution due to timeout. This limits the number of iterations to 25 for evaluations. It can be observed from Figures 2 and 3 that the advantages of approach GLNR-SP grow for larger network sizes, that is, it exhibits a lower runtime growth with increasing VN and SN sizes. In case of embedding VN with 50 nodes to SN with 1000 nodes, when the networks reach their respective maximal sizes, GLNR-SP's runtime is around 1.04 seconds, compared with around 2.03 seconds, 502.94 seconds, 1788.84 seconds, 1.80 seconds, 2585.31 seconds, and 1.04 seconds for its counterparts GAR-SP, DViNE-SP, RW-MM-SP, vnmFlib, VNE-NTANRC, and k-core, respectively. Comparison of embedding quality leads to similar results, as listed in Table 6 and depicted in Figure 3. The results are indicative of proposed method's suitability for application to scenarios in which SN is large-scale.

5.3.5 | Explanation of experimental results

In summary, experimental results presented in this section indicate that algorithm GLNR-SP might efficiently implement scalable VNE with enhanced quality in terms of the main evaluational metrics, particularly in case of the massive SN. It can be observed that among four metrics of evaluating VNE algorithms, namely, runtime, VNR acceptance ratio, C/R, and NUR, GLNR-SP has earned a significant promotion in runtime when SN becomes massive. Such an improvement originates from that procedure GenSubset was implemented with low computational overhead to pick out a subgraph S prior to performing the procedure LocalRank, and generally S is far smaller than G and capable of fulfilling the embedding of H with a high probability. Relative to GLNR-SP, its counterparts for comparison execute combinatorial search or iteration convergence on SN to choose the nodes and links that fulfill the demands from H, which usually consumes the most of time overhead. Moreover, GLNR-SP brings on a high VNR acceptance ratio because it associates virtual and substrate nodes more accurately when performing LocalRank procedure. Moreover, by algorithm GLNR-SP, no evident degradation emerges in the rest of metrics: C/R, and NUR because the resource available and topological association have been taken into account in stage of generating S. Exceptionally, optimization-based method DViNE-SP cannot compete the task of embedding for SNs larger than 1000, given that it already reaches their computational limits before the end of the simulation range.

Experimental results are obtained under preconditions that hypotheses H1, H2, and H3 have been accepted. Namely, first of all, a VN link is mapped to a length-2 at most shortest path. Then, the numbers of SN and VN nodes increase according to $n = 5 \times i$ and $m = 2 \times i$, respectively. Finally, node resource and link width of SN and VN are subject to the uniform distribution. Acceptance/rejection of these hypotheses imposes an impact to experimental results mainly on two evaluational metrics: runtime and AR. If hypothesis H1 is rejected, that means path splitting should be considered, the link mapping generally requires extra runtime to estimate the residual bandwidth between substrate node pairs. In addition, if hypothesis H2 is rejected, the scales of SN and VN may grow more rapidly with the iteration i of experiment. If both H1 and H2 are rejected, the runtime of all algorithms will increase significantly. In case of rejecting H3, more VN requirements will be rejected due to nonuniform resource distribution in interval [min, max], which generally leads to a lower AR.

Approach in this research relies heavily on techniques of local random walking on substrate graph to associate virtual nodes to substrate nodes. There is significant difference between GLNR-SP and previously known graph matching algorithms, because the VNE problem requires a correspondence between nodes, but a link in VN can be mapped to a path in SN, this being a one-to-multiple relation that is not strictly a mathematically defined “mapping.” A direct application of known methods for node mapping might lead to a low computational efficiency and a poor VNR acceptance ratio, such as the GAR-SP greedily provisioning available resource, the vnmFlib based on graph matching, the RW-MM-SP and VNE-NTANRC ranking nodes with topology properties. The low computational efficiency of previously known heuristic VNE methods is mainly caused by that they experience a process of either combinatorial search (DViNE-SP) or iterative convergence (RW-MM-SP and VNE-NTANRC). A key advantage of the scheme proposed in this paper, compared with previously known heuristic VNE methods, is that it is not iterative. This will avoid the process of random walk converging to the stationary distribution such that a better balance between efficiency and quality has been achieved.

6 | CONCLUSION

As a substantial component in network virtualization technology, VNE has received considerable attention from the research community and communication industry. Heuristic algorithms are increasingly sought for overcoming the computational intractability of optimization-based approaches. However, it is an extremely challenging goal to achieve a solution that is acceptable to both users and providers, which usually means short response time and high quality. This goal becomes even more critical when SN scales to large sizes. In this case, the term "acceptance" means not only that a VNR can be technically fulfilled but also users agree with the embedding efficiency and quality, which implies that efficiency should be an even more essential factor to affect user's decision in the current user-centered environment.

This work addresses the problem of massive VNE by decomposing of the general process of node ranking into two procedures: global and local node ranking using local random walk based on discrete Green's function. The proposed scheme has been presented both theoretically and algorithmically, and a series of simulations has been conducted to confirm the theory-based intuition that the proposed scheme offers benefits in runtime and embedding quality for a general VNE task.

The fact that experimental work is conducted on Hub-Star network limits the applicability of the experimental verification. The authors plan to extend the experimental scenarios to other kinds of topologies, including real world networks. Theoretically, the authors are also interested in exploring more accurate strategies for global node ranking in extremely large-scale VN and SN.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China under grant 61562089.

AUTHOR CONTRIBUTIONS

Chenggui Zhao performed the theoretical derivation, experiments, and wrote the paper. Behrooz Parhami checked the derivation and revised the paper.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

ORCID

Chenggui Zhao https://orcid.org/0000-0002-1570-0435

REFERENCES

1. Fischer A, Botero JF, Beck MT, De Meer H, Hesselbach X. Virtual network embedding: a survey. *IEEE Commun Surv Tutor*. 2013;15(4):1888-1906.
2. Li J, Zhang N, Ye Q, Shi W, Zhuang W, Shen X. Joint resource allocation and online virtual network embedding for 5G networks. In: Proceedings of the IEEE Global Communications Conference (GLOBECOM); 2017; Singapore.
3. Cheng X, Su S, Zhang Z, et al. Virtual network embedding through topology-aware node ranking. *ACM SIGCOMM Comput Commun Rev*. 2011;41(2):38-47.
4. Luizelli MC, Bays LR, Buriol LS, Barcellos MP, Gaspary LP. How physical network topologies affect virtual network embedding quality: a characterization study based on ISP and datacenter networks. *J Netw Comput Appl*. 2016;70:1-16.
5. Chowdhury M, Rahman MR, Boutaba R. Vineyard: virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Trans Netw*. 2012;20(1):206-219.
6. Mohar B, Poljak S. Eigenvalues in combinatorial optimization. In: *Combinatorial and Graph-Theoretical Problems in Linear Algebra*. Berlin, Germany: Springer; 1993:107-151.
7. Tsiatas A, Saniee I, Narayan O, Andrews M. Spectral analysis of communication networks using Dirichlet eigenvalues. In: Proceedings of the International Conference on World Wide Web (WWW); 2013; Rio de Janeiro, Brazil.
8. Chung F. Laplacians and the Cheeger inequality for directed graphs. *Ann Comb*. 2005;9(1):1-19.
9. Chung F. Random walks and local cuts in graphs. *Linear Algebra Appl*. 2007;423(1):22-32.
10. Chung F. PageRank as a discrete Green's function. *Geom Anal I ALM*. 2010;17:285-302.
11. Chung F, Yau S-T. Coverings, heat kernels and spanning trees. *Electron J Comb*. 1998;6(1):163-184.
12. Chung F, Yau S-T. Discrete Green's functions. *J Comb Theory A*. 2000;91(1-2):191-214.
13. Jarray A, Karmouch A. Decomposition approaches for virtual network embedding with one-shot node and link mapping. *IEEE/ACM Trans Netw*. 2015;23(3):1012-1025.
14. Chen X, Li C, Jiang Y. Optimization model and algorithm for energy efficient virtual node embedding. *IEEE Commun Lett*. 2015;19(8):1327-1330.

15. Yu M, Yi Y, Rexford J, Chiang M. Rethinking virtual network embedding: substrate support for path splitting and migration. *ACM SIGCOMM Comput Commun Rev*. 2008;38(2):17-29.
16. Lischka J, Karl H. A virtual network mapping algorithm based on subgraph isomorphism detection. In: Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures; 2009; Barcelona, Spain.
17. Zhao C, Pu Z. Recent advances and trends in virtual network embedding. *IEICE Trans Commun*. 2016;99(6):1265-1274.
18. Fan W, Li J, Ma S, Wang H, Wu Y. Graph homomorphism revisited for graph matching. *Proc VLDB Endow*. 2010;3(1-2):1161-1172.
19. Zhang D, Gao L. Virtual network mapping through locality-aware topological potential and influence node ranking. *Chin J Electron*. 2014;23(1):61-64.
20. Cao H, Yang L, Zhu H. Novel node-ranking approach and multiple topology attributes-based embedding algorithm for single-domain virtual network embedding. *IEEE Internet Things J*. 2018;5(1):108-120.
21. Cao Y, Fan W, Ma S. Virtual network mapping: a graph pattern matching approach. In: *Data Science: 30th British International Conference on Databases, BICOD 2015, Edinburgh, UK, July 6-8, 2015, Proceedings*. Cham, Switzerland: Springer International Publishing; 2015.
22. Beck MT, Fischer A, Botero JF, Linnhoff-Popien C, de Meer H. Distributed and scalable embedding of virtual networks. *J Netw Comput Appl*. 2015;56(C):124-136.
23. Zhang S, Qian Z, Wu J, Lu S, Epstein L. Virtual network embedding with opportunistic resource sharing. *IEEE Trans Parallel Distrib Syst*. 2014;25(3):816-827.
24. Wang X, Song M, Yuan D, Liu X. Robust virtual network embedding based on component connectivity in large-scale network. *China Communications*. 2017;14(10):164-179.
25. Chung F, Tsiatas A, Xu W. Dirichlet pagerank and trust-based ranking algorithms. In: *Algorithms and Models for the Web Graph: 8th International Workshop, WAW 2011, Atlanta, GA, USA, May 27-29, 2011. Proceedings*. Berlin, Germany: Springer; 2011. *Lecture Notes in Computer Science*; vol. 6732.
26. Kolev P, Sun H. Dirichlet eigenvalues, local random walks, and analyzing clusters in graphs. In: *Algorithms and Computation: 25th International Symposium, ISAAC 2014, Jeonju, Korea, December 15-17, 2014, Proceedings*. Cham, Switzerland: Springer International Publishing; 2014.
27. Beck MT, Linnhoff-Popien C, Fischer A, Kokot F, De Meer H. A simulation framework for virtual network embedding algorithms. In: Proceedings of the 16th International Telecommunications Network Strategy and Planning Symposium (Networks); 2014; Funchal, Portugal.

How to cite this article: Zhao C, Parhami B. Virtual network embedding on massive substrate networks. *Trans Emerging Tel Tech*. 2020;31:e3849. https://doi.org/10.1002/ett.3849