Virtual Network Embedding Through Graph Eigenspace Alignment

Chenggui Zhao and Behrooz Parhami, *Life Fellow, IEEE*

***Abstract*—The purpose of this paper is to study tradeoffs between efficiency and precision for the general virtual network embedding (VNE) problem. To narrow the gap between solutions contributed by previous heuristic schemes and the optimal solution, and to ease the unacceptable computational burden of optimization approaches to VNE for large-scale networks, we propose the use of graph eigenspace notions for node mapping, that is, for associating virtual nodes to substrate nodes (SNs). We also contribute an inexact algorithm which projects all SNs to a 2-D eigenspace for generating a more efficient node mapping. There are some similarities between our method and numerous graph-matching strategies emerging in machine learning fields, but there are also some differences between the two, because the VNE problem is not strictly mathematical with respect to the mapping it seeks. Thus, we provide the relevant theoretical evidence to guarantee the solution quality of our scheme compared with a couple of previous VNE proposals. Simulation results demonstrate that our schemes can reach a better tradeoff point regarding runtime (that approaches the overhead of implementing eigendecomposition of a general matrix) and embedding quality metrics. Besides, our inexact algorithm using 2-D projection exhibits an improvement in quality over previous node-rank-based methods.**

***Index Terms*—Virtual network embedding, distance matrix, eigenspace decomposition.**

I. Introduction

Network virtualization constitutes a promising technology that separates the traditional network-level functions of currently encapsulated Internet architecture from Internet service providers (ISPs), by splitting the function offered by ISPs into parts of infrastructure and service providers. Regardless of the business models used in network virtualization, ISPs equipped with virtualization can be abstracted as integration of infrastructure providers (InPs) in charge of disposing the physical network resources and virtual network providers (VNPs) to provision virtual resources from InPs in order to respond to virtual network requests (VNR) from users. This flexible separation allows users to share a virtualized network environment while ignoring the concrete underlying implementation and resource scheduling. Obviously, there should be an intermediate module between VNP and InPs for mapping network resources requested to physical network entities such that ISPs can sufficiently utilize network resources to provide user-customized network services. This module has been generally referred to as virtual network embedding (VNE), mapping, or provisioning [1]. Figure 1 offers an intuitive view of the two-level architectural model for virtual network embedding just described.

As a central part of network virtualization, virtual network embedding has attracted much attention from the network research community. The significant volume of work has led to the emergence of a variety of representation forms. For our purposes, we categorize current VNE approaches into two groups: mathematical optimization-based representations(such as those of [2]–[4]), and graph models favored by a majority of heuristic strategies (e.g., [1] and [5]–[7]). VNE problems can be categorized as static or dynamic, according to whether VN is configured for its lifetime. More detail can be found in a recently published survey [1]. Table I provides a list of key abbreviations used.

A. VNE Solutions Based on Mathematical Optimization

Optimization-based VNE proposals formulate VNE as a problem of combinational optimization, then solve it by linear or integer programming to discover an exact solution. Chowdhury *et al.* [3] propose to handle the correlation between node and link embedding phases. They extend the substrate network to an augmented substrate graph in which they create a meta node for each virtual node, and each meta node is connected to all substrate nodes in a cluster containing the substrate nodes within some distance to the virtual node corresponding to this meta node, by a meta edge with infinite bandwidth. Then, the VN embedding problem is formulated as a mixed-integer program via substrate network augmentation, and two online VN embedding algorithms, deterministic D-ViNE and randomized R-ViNE, are devised for solving the linear program with relaxed integer constraints.

Jarray and Karmouch [2] propose a decomposition approach termed A-JNLE CG to realize one-shot node and link mapping, and to cause competition among VN users. The scheme's main idea is to decompose an overall VNE problem into a main problem with constraints on the availability of substrate

Manuscript received June 8, 2018; revised October 15, 2018 and December 23, 2018; accepted January 13, 2019. Date of publication January 25, 2019; date of current version June 10, 2019. Chenggui Zhao's work was supported by the National Natural Science Foundation of China, Grant No. 61562089. The associate editor coordinating the review of this paper and approving it for publication was Q. Ling. *(Corresponding author: Chenggui Zhao.)*

C. Zhao is with the School of Information, Yunnan University of Finance and Economics, Kunming 650221, China, and also with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106 USA (e-mail: zhaochenggui@126.com).

B. Parhami is with the Electrical and Computer Engineering Department, University of California at Santa Barbara, Santa Barbara, CA 93106 USA (e-mail: parhami@ece.ucsb.edu).

Digital Object Identifier 10.1109/TNSM.2019.2895354

TABLE I
KEY ABBREVIATIONS

InP	Infrastructure Provider
ISP	Internet Service Provider
SN	Substrate Network
VN	Virtual Network
VNE	Virtual Network Embedding
VNP	Virtual Network Provider
VNR	Virtual Network Request

resources and a pricing problem with constraints on the embedding of VN requests. A-JNLE CG relies on column generation technology (CG), which appends an additional column to the constraint matrix of the primary object. Furthermore, multiple VN requests can be processed through iterative generation of columns so as to reduce blocking among them. In addition, competition is stimulated among multiple VNRs through a periodical auction to allocate substrate resources.

The optimization objectives contained in most known VNE schemes involve parameters of particular interest to users and VNPs, such as maximal provider revenue and acceptance ratio, and minimal embedding cost (see [2], [3]). Particularly, the acceptance ratio of an VNE algorithm indicates the proportion of virtual network requests successfully accepted by implementing that algorithm over a period T, and can be defined by $\sum_{t=0}^{T} n_a / \sum_{t=0}^{T} n_r$, where n_a is the number of accepted VN requests and n_r is the number of total VN requests. There are also proposals with concerns for other factors. For instance, a minimization model of energy-efficient virtual node embedding was constructed and solved to yield the minimal product of energy [4].

B. VNE Solutions Based on Topological Correspondence

Finding the optimal VNE solution via optimization methods leads to high computational complexity, as the problem is NP-hard. This becomes evident when a general graph embedding with constraints is modeled as the multiway separator problem assigning virtual nodes to the substrate nodes meeting constraints of the node capacities and link bandwidths. It is also apparent when we treat VNE as seeking a p-homomorphism between a graph H and a subgraph of other graph G. Defining p-homomorphism formally, Fan *et al.* [8] have proven that it cannot be computed in polynomial time.

Given the NP-hardness of VNE, much research has dealt with designing heuristic algorithms to solve it [1], [5]–[7]. These algorithms usually achieve a near-optimal solution, i.e., their results fall between approximate and optimal solutions. Yu *et al.* [5] simplify virtual link embedding by relaxing the problem solution space. In this approach, substrate network splits substrate path, associating a virtual link to subpaths, which can be periodically migrated to promote the utilization of substrate resource and ultimately obtain a good solution. Algorithmically, they aim for maximal resource utilization in the substrate network. Hence the virtual nodes are greedily mapped to the substrate nodes with the maximum amount of substrate resources so as to minimize the use of the resources at the nodes with less available resources. They then map virtual links to shortest paths spanning the substrate nodes mapped to its two incident nodes in the substrate network.

Topology-aware VNE means that topological features of VN and SN are taken into account in discovering the correspondence of nodes and links in the two networks. Cui *et al.* [9] propose a VNE algorithm based on topology convergence degree, using maximal topology convergence degree to ensure that proximity in VNE is maintained when a VN is mapped to SN. Balanced ratio on links is defined for load balancing on SN.

The correspondence between VN and SN nodes is deemed more efficient if nodes of both networks can be ranked according to some measure that reflects their topological features. Cheng *et al.* [7] consider topological properties of SN nodes as important VNE parameters in the node-mapping stage. Inspired by Google's PageRank algorithm, they rank all virtual and substrate nodes according to their relative importance. The relative importance α of each node is given by $\alpha = \alpha_0 + \alpha_j + \alpha_f$, where α_0 is the product of node CPU cycle and the sum of bandwidths on outgoing links, α_j (jump probability) represents the weighted importance of all reachable nodes, and α_f (forward probability) estimates the weighted importance of out-neighbors. Then, virtual and substrate nodes are sorted separately in decreasing order of node ranks. Consequently, higher-ranked virtual nodes have priority for being mapped to higher-ranked substrate nodes. Mapping of virtual links is done by shortest-path algorithm if there is no path splitting or by multi-commodity flow algorithm if path splitting is permitted. Node association was added to the VNE algorithm to improve efficiency and acceptance ratio.

Analogous to [7], but ranking nodes in different way, Zhang and Gao [10] apply a topological potential function to measure and rank the topology importance of SN nodes, realized by an algorithm called locality-aware node topological potential ranking (LNTPR). They consider the mutual influence between a VN node and its objective nodes, and incorporate the concept of locality awareness into LNTPR to get an improved topology-aware VNE algorithm named locality-influenced choice of node (LICN).

Lischka and Karl [6] offer a VNE algorithm by detecting subgraph isomorphism between topologies of VN and SN to discover the correspondence between both nodes and links in the same stage. Evaluations show that VNE based on subgraph isomorphism detection is particularly efficient compared to the two-stage approach for large VNs with high resource demands.

Another way to find the topological correspondence of nodes and links between VN and SN is through pattern matching. Proposals based on graph clustering rely mostly on graph pattern matching to discover the node mapping. Reference [11] treats VN and SN as a constrained graph and weighted graph, respectively, with weights on nodes and edges indicating their capacities. A variety of VNE constraints can be described in this model, so as to formulate VNE as an optimization problem with a mapping cost function. Adhering to this model, it is approximation-hard to solve the intractable optimization problems for various mapping constraints.

C. Other Approaches to the VNE Problem

In multi-domain or cross-domain virtual network embedding (MVNE) problem, SN resources may be configured and provisioned from multiple InPs. MVNE can be decomposed into a set of single-domain VNEs. Esposito *et al.* [12] institute a general consensus-based auction mechanism to tackle the distributed VNE problem. Their algorithm exchanges bids information of substrate nodes with their neighbors to discover the available resources. Then it iterates over bidding, and reaches consensus when a solution is found. Distributed VNE policies are developing along with multi-domain approaches, although they are not essential to the latter. Beck *et al.* [13] design a distributed and parallel framework called DPVNE to implement a VNE, in which several VNE algorithms are run to distributively map VNs into SN, so that the single point pressure in SN is reduced and greater efficiency is achieved.

VNE strategies minimizing energy cost are critical for InPs, since energy consumption accounts for more than half of the total cost of the substrate networks. Multi-objective integer linear programming has been applied to accommodate as many VNRs as possible simultaneously, so as to maximize the revenue of InPs, and to consolidate VNs into the minimum number of substrate nodes to minimize the energy consumption [14]. Subsequently, VNE solution is obtained via an algorithm based on artificial immune system.

Zhang *et al.* [15] contribute ORS, an opportunistic resource-sharing scheme to deal with time-dependent VNRs. This scheme decouples the variable section from required resources, and expresses the time-slot assignment as an optimization process analogous to the well-known bin packing problem. Two solutions based on the notion of first-fit were proposed to allocate time slot, which enables mapping the bandwidth of virtual links to corresponding time slots so as to realize multiple VNs while sharing substrate resources.

D. Outline of Our VNE Approach

It is known that a class of combinatorial optimization problems, such as node ranking and pattern matching, can be simplified with graph eigenspace techniques, on account of calculating the eigenpairs of a graph consuming far less time than advanced combinatorial search procedures in large objective spaces. The motivation behind advocating eigenspace techniques to solve graph-representation applications arises from the fact that the magnitude and distribution of eigenvalues and eigenvectors of a graph substantially reflect its structural and geometric properties. The most successful application of the graph eigenspace paradigm might be Google's PageRank, which exploits eigenvectors belonging to the largest eigenvalues of the adjacency matrix of the graph indicating links between Web pages returned from key search to define a certain desirable ordering of selected Web pages.

Subsequently, analogous methods were widely applied in multiple domains to discover a hidden goal, usually expressed as subgraphs in a given graph pattern. This approach is known as subgraph matching [16] or network alignment [17]. Knossow *et al.* [16] formulate inexact large and sparse graph matching with decomposition and dimensionality reduction of the graph eigenspace. Singh *et al.* [17] apply network alignment for identifying possible mappings between the nodes representing the proteins of protein networks to understand how proteins in the cell interact. This work assumes that two nodes should be associated with high possibility if their respective neighbors are well matched, with node matching possibility expressed as the accumulation of those of its neighbors. If we treat VNE as the problem of node assignment or matching between two graphs, schemes emulating PageRank can be exploited for dealing with virtual network embedding problem. This has been implemented in [7], in a manner analogous to the node ranking scheme of [17].

Our approach relies heavily on graph eigenspace techniques to associate virtual nodes to substrate nodes. We also contribute an inexact algorithm to rank these nodes for more efficient mapping. There is significant difference between our method and previously known graph matching algorithms, because the VNE problem requires node mapping (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 based on spectral graph theory matching for node mapping might lead to a poor VNR acceptance ratio. Whereas, our algorithm for inexact node ranking resembles that of [7], our algorithm is not iterative. This will avoid the process of random walk converging to the stationary distribution.

Heuristic VNE algorithms (or other assignment and scheduling strategies) are judged by how close they come to optimal solutions. It is often the case for heuristic schemes that the closeness of results to optimal solutions is known only in a statistical sense, is established via unsatisfactorily loose bounds, or is judged by simulation results. This is problematic, as there may be worst-case problem instances that deviate significantly from the expected behavior. A key advantage of the scheme we present in this paper compared with previously known heuristic VNE methods is that the eigenspace-based approach can provide theoretical guarantees for the solution quality.

II. Virtual Network Embedding Formulation

For the sake of ready reference, we list the mathematical notation used in our formulation of VNE in Table II.

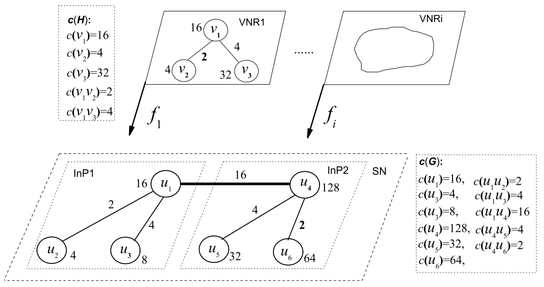
A. Substrate Network and Virtual Network Request

Substrate network 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. If the existence of r types of node or link constraints is contemplated, where "constraint" refers to resource or demand attributes, depending on whether the described object is substrate or virtual network, respectively, then each entity x should be identified as an r-tuple $c(x) = (c_1(x), c_2(x), \ldots, c_r(x))$, where the functions $c_i(x)$ represent the magnitude of the constraint at network entity x.

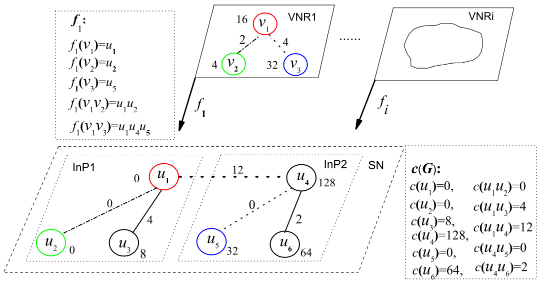
As an example, Fig. 1a shows an example virtual network consisting of nodes v_1, v_2, v_3 and links v_1v_2, v_1v_3, which is to be embedded to a substrate network with nodes u_1, u_2, u_3

TABLE II
MATHEMATICAL NOTATION

G	Substrate network graph
H	Virtual network graph
$f(h) \to g$	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
$c(g \to h)$	The resource in entity g mapped to h
$\sigma(f)/r(f)$	Cost/revenue of mapping f
$p(u, v)$	Shortest path between node u and v
$E(p)$	Set of links on path p
A_X/W_X	Standard/weighted adjacency matrix of graph X
D_X	Distance matrix of graph X
$\lambda_i(X)/q_i(X)$	Eigenvalues / eigenvectors of matrix X
Q_X	Matrix with columns being eigenvectors of X
Λ_X	Diagonal matrix with eigenvalues of X as diagonal entities
$X(i : j)$	Columns submatrix of X with subscripts ranging from i to j



(a) Embedding VNR1 to SN by f_1



(b) VNR1 Embedded to SN by f_1

Fig. 1. A two-level example architectural model for virtual network embedding, with the correspondence between virtual edge and physical edge for InP2 omitted, numbers near circles indicate switching capacity of routing/switching devices, and the ones near the links represent transmission bandwidth, in Gbps. Virtual-to-physical node correspondence is marked by distinct colors, and edge correspondence is marked by distinct dotted and dashed lines.

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 $c(u_1u_2) = 4$ and $c(v_1v_2) = 2$, respectively.

B. Virtual Network Embedding

The problem of embedding H into G can be modelled as finding a mapping $f : H \to G$, consisting of node and link mappings, described as:

Node Mapping: each virtual node $v_h \in V(H)$ is embedded to a different node $v_g \in V(G)$ by $f(v_h) = v_g$ which defines the virtual node mapping from $V(H)$ to $V(G)$;

Link Mappings: each virtual link $u_hv_h \in E(H)$ in a virtual network is embedded to a substrate path $p(f(u_h), f(v_h))$ by $f(u_hv_h) = p(f(u_h), f(v_h))$ which defines the virtual link mapping from $E(H)$ to $E(G)$.

The mapping f from H to G must satisfy the constraint conditions: $\forall h \in V(H)$ or $h \in E(H)$, $c(h) \leq c(f(h))$.

Fig. 1b shows that a virtual network has been embedded to a substrate network by the mapping $f_1 : H \to G$, through which the virtual nodes v_1 and v_2 have been embedded to two physical nodes u_1 and u_2 successfully, without violating the constrains $c(v_1) = c(u_1) = 16$ and $c(v_2) = c(u_2) = 4$. However, virtual nodes v_3 cannot initially be embedded to physical nodes u_3 initially due to the resource mismatch $c(v_3) = 32 > c(u_3) = 8$. Therefore, another candidate node u_5 has been chosen for embedding v_3 because it does satisfy the resource constraint $c(v_3) = c(u_5) = 32$. Subsequently, link mappings have been executed as $f_1(v_1v_2) = u_1u_2$, $f_1(v_1v_3) = u_1u_4u_5$ successfully due to $c(v_1v_2) = c(u_1u_2) = 4$ and two links u_1u_4, u_4u_5 on path $u_1u_4u_5$ have bandwidth constraints $c(u_1u_4) = 16$ and $c(u_4u_5) = 4$ that meet the bandwidth demands $c(v_1v_3) = 4$. Consequently, node and link resources in G have been occupied partially due to embedding of H; this can be readily observed by noting that $c(u_1) = c(u_2) = c(u_5) = c(u_4u_5) = 0$ and that $c(u_1u_4)$ has been reduced from 16 to 12.

A concern can be raised from the above example depicted in Fig. 1. It is quite obvious that there exist multiple potential mappings f for embedding a virtual network H and it is not clear which one is optimal with respect to a user's objectives? Such a question is what motivates an optimization formulation for describing the VNE problem defined previously.

As described in previously cited VNE references [1]–[3], [5]–[7], the VNE objective is to devise the embedding f so as to utilize SN resources as fully as possible. Let z denote the optimization objective, then the VNE problem can be characterized as solving the following optimization problem:

$$\arg\min_f \left\{ z(f) \middle| z(f) = \sum_{l_h \in E(H)} \sum_{l_g \in E(f(l_h))} z(c(l_g)) + \sum_{v_h \in V(H)} \sum_{v_g \in f(v_h)} z(c(v_g \to v_h)) \right\}. \tag{1}$$

Previous works [1]–[3], [5]–[7], using f to embed a virtual network H to substrate network G, are driven mainly by two optimization criteria: revenue $r(f)$ and embedding cost $\sigma(f)$. These two objectives can be formulated as

$$r(f) = \sum_{v_h \in V(H)} c(v_h) + \sum_{l_h \in E(H)} c(l_h), \tag{2}$$

$$\sigma(f) = \sum_{v_h \in V(H)} c(v_h) + \sum_{l_h \in E(H)} \sum_{l_g \in E(f(l_h))} c(l_g \to l_h). \tag{3}$$

III. Eigenspace-Based Schemes

We are motivated to use graph eigenspace technology to solve the VNE problem in light of the rising interest in the application of graph eigenspace in handling combinatorial optimization problems. One of ways of applying eigenvalues to combinatorial optimization is to use the eigenvalues to transform combinatorial optimization problems to corresponding continuous optimization problems. A relevant survey by Zhang [19] has revealed that the graph eigenspace based approach leads to the solution of some problems with excellent tradeoff in quality and efficiency, the two often-used competing criteria in combinatorial optimization. For example Beck *et al.* [22] used the eigenvalue approach in the study of the quadratic assignment and graph partition problems.

In the rest of this section, we first present a coarse-grained optimization formula used in the eigenspace-based scheme for embedding static and dynamic VNs. Then, we propose a family of algorithms, involving multiple relevant distance matrices, based on this formula. We begin by transforming the optimization formulation for VNE problem to a form of graph path distance matrix, and describe the process of solving it. Then, we improve the quality of solution via the alignment of eigenspaces of virtual and substrate networks. Finally, we consider several generalized distance measures that lead to a family of VNE algorithms called DMEA-X (*X* standing for the distance matrix used in algorithm). Also, for computationally efficient VNE, an inexact algorithm called distance matrix eigenspace alignment with 2D projection (DMEA-2D) is devised.

A. Connection of the VNE Optimization Formula to the Unweighted Distance Matrix of Substrate Network

We first build a connection between the VNE optimization formula and the unweighted distance matrix of substrate network such that the VNE solutions are presented in terms of eigenpairs of this matrix.We denote the standard adjacency matrix of graph X as A_X, and the weighted one as W_X representing the weights distribution of entities on graph *X*. Both A_X and W_X can model the communication topology of graph *X*. We also need a set of distance metrics in matrix form deduced from A_X or W_X, and denote them as D_X in either case, to formulate the VNE process. The substrate network graph *G* = (*V*(*G*), *E*(*G*)) with its standard adjacency matrix A_G is defined by:

if $(u_g, v_g) \in E(G)$ then $A_G(u_g, v_g) = 1$, else $A_G(u_g, v_g) = 0$.

The virtual network graph *H* = (*V*(*H*), *E*(*H*)) is represented by its weighted adjacency matrix W_H, with entities defined by:

if $(u_h, v_h) \in E(H)$ then $W_H(u_h, v_h) = c(u_h v_h)$, else $W_H(u_h, v_h) = 0$.

To address the VNE optimization formulation, the all-pairs shortest distance (APSD) matrix should be computed a priori from the standard adjacency matrix A_G. Let D_G denote the all-pairs unweighted shortest path matrix. Matrix D_G has nonzero entries $D_G(u_g, v_g)$ if and only if there is a connected path $p(u_g, v_g)$ so that $D_G(u_g, v_g)$ is equal to the length of unweighted shortest path between u_g and v_g. Thus, the (u_g, v_g)-entry of matrix D_G is defined as:

$$D_G(u_g, v_g) = |p(u_g, v_g)|.$$

Shifting attention to the generic VNE optimization formulated in (1), we can write the optimization expression (1) as

$$\arg\min_f \{z_v + z_l\},$$

by separating the optimization into two parts:

$$z_v = \sum_{v_h \in V(H)} \sum_{v_g \in f(v)} z(c(v_g \to v_h)), \text{ and}$$

$$z_l = \sum_{l_h \in E(H)} \sum_{l_g \in E(f(l_h))} z(c(l_g)).$$

Since v_h can only be mapped to one node v_g, v_g must assign minimum resource $c(v_h)$ to virtual node v_h. Thus, we have $z_v = \sum_{v_h \in V(H)} c(v_h)$, which means z_v is independent of the mapping *f*. So, the optimization objective can be formulated as:

$$\arg\min_f \left\{ z(f) | z(f) = \sum_{l_h \in E(H)} \sum_{l_g \in E(f(l_h))} z(c(l_g)) \right\}. \quad (4)$$

If the constraint $c(l_h)$ on a virtual link $l_h = (u_h, v_h)$ is afforded by a substrate path $p(f(u_h), f(v_h))$, then each link lying on $p(f(u_h), f(v_h))$ will contribute a resource quantity $c(l_h)$ for occupying the virtual link l_h. This means that

$$\begin{aligned} z(f) &= \sum_{l_h \in E(H)} \sum_{l_g \in E(f(l_h))} z(c(l_g)) \\ &= \sum_{l_h \in E(H)} \sum_{l_g \in E(f(l_h))} c(l_g \to l_h) \\ &= \sum_{l_h \in E(H)} |f(l_h)| c(l_h) \\ &= \sum_{l_h \in E(H)} |E(p(f(u_h), f(v_h)))| c(l_h), \end{aligned} \quad (5)$$

where $|f(l_h)|$ denotes the number of links on the path $f(l_h)$. Equation (5) can be further expressed as

$$\begin{aligned} z(f) &= (1/2) \sum_{u_h \in V(H)} \sum_{v_h \sim u_h} D_G(f(u_h), f(v_h)) c((u_h, v_h)) \\ &= (1/2) tr(D_G(f(H), f(H)) W_H), \end{aligned} \quad (6)$$

where *tr*(*X*) represents the trace of matrix *X* and *D*(*f*(*H*),*f*(*H*)) is the submatrix of D_G whose rows and columns are indexed by *f*(*H*) defined as $f(H) = \{f(v_h) | v_h \in E(H)\}$. Without loss of generality, we assume that the permutation matrix *P* permutes the *m* rows $D_G(f(H), :)$ of D_G but keeps the other rows unchanged, and it permutes the columns likewise. Then, the virtual network mapping *f* can be connected to a permutation matrix *P* through

$$D_G(f(H), f(H)) = [P D_G P^T]_{1:m},$$

where $X_{1:m}$ is the principal submatrix ranging from 1 to *m* of *X*. Combining with (6), we obtain an optimization formulation extremely close to the well-known quadratic assignment problem (QAP) as follows:

$$\arg\min_P\{z(P)|z(P) = (1/2)tr([PD_GP^T]_{1:m}W_H)\}. \quad (7)$$

If the eigenvalues of $[PD_GP^T]_{1..m}$ and W_H, shown in increasing order are $\lambda_1(D_G),\ldots,\lambda_m(D_G)$ and $\lambda_1(W_H),\ldots,\lambda_m(W_H)$, respectively, the optimization above can be bounded as

$$tr([PD_GP^T]_{1..m}W_H) \geq \sum_{1\leq i\leq m}\lambda_i(W_H)\lambda_{n-i+1}(D_G) \quad (8)$$

given the Hoffman-Wielandt inequality, which asserts that:

$$tr(Y(PXP^T)) \geq \sum_{1\leq i\leq n}\lambda_i(Y)\lambda_{n-i+1}(X). \quad (9)$$

For symmetric D_G and W_H, the lower bound can be realized by $P = Q_GQ_H^T$ (see [18, ch. VI, pp. 152–181]), where Q_G and Q_H are matrices consisting of orthogonal eigenvectors of D_G and W_H, respectively. Note that our inequality (8) is slightly different from (9). Thus, we need the following theorem to guarantee that there exists an analog of $P = Q_GQ_H^T$ as the closed-form solution of our proposed VNE optimization described in model (7). Let $X(i:j)$ mark a submatrix consisting of columns from X by selecting ascending $i \leq j$/descending $i \geq j$ subscripts from i to j, namely, $X(i{:}j) = [x_i, x_{i-1},\ldots,x_j]$ when $i \geq j$ and $X(i{:}j) = [x_i, x_{i+1},\ldots,x_j]$ when $i \leq j$.

Theorem 1: If $(\lambda_i(D_G),\ q_i(D_G))$ $(1 \leq i \leq n)$ and $(\lambda_j(W_H),\ q_j(W_H))$ $(1 \leq j \leq m)$ are eigenpairs of matrices D_G and W_H in increasing eigenvalues order, respectively, the VNE optimization formulation

$$\min_P\left\{tr([PD_GP^T]_{1:m}W_H)\right\} \quad (10)$$

reaches the lower bound $\sum_{1\leq i\leq m}\lambda_i(W_H)\lambda_{n-i+1}(D_G)$ when $P = [Q_G(n:n-m+1)Q_H^T, Q_G(n-m:1)]$, where $Q_G(i:j)(i \geq j) = [q_i(D_G), q_{i-1}(D_G),\ldots, q_{i-j+1}(D_G)]$, and $Q_H = [q_1(W_H), q_2(W_H),\ldots,q_m(W_H)]$.

Proof: Firstly, to apply inequality (9), we transform optimization form (10) to the following blocked form:

$$\begin{aligned} &tr([PD_GP^T]_{1:m}W_H)\\ &= tr\left(\begin{bmatrix}(P^TD_GP)_{1..m} & *_{12}\\ *_{21} & *_{22}\end{bmatrix}\begin{bmatrix}W_H & 0\\ 0 & 0\end{bmatrix}\right)\\ &= tr\left(P^TD_GP\begin{bmatrix}W_H & 0\\ 0 & 0\end{bmatrix}\right), \end{aligned} \quad (11)$$

where "$*_{ij}$" means a matrix block located in position $(i,\ j)$ without regard to its actual entry values. Now, applying inequality (9) to (11), the lower bound can be transformed to:

$$\begin{aligned}&\min_P\left\{tr\left(P^TD_GP\begin{bmatrix}W_H & 0\\ 0 & 0\end{bmatrix}\right)\right\}\\ &\geq \sum_{1\leq i\leq m}\lambda_i(W_H)\lambda_{n-i+1}(D_G).\end{aligned} \quad (12)$$

Instantiating the permutation matrix P as the orthogonal matrix $P = [Q_G(n:n-m+1)Q_H^T, Q_G(n-m:1)]$, further computation reveals:

$$\begin{aligned}&tr\left(P^TD_GP\begin{bmatrix}W_H & 0\\ 0 & 0\end{bmatrix}\right)\\ &= tr\left(\begin{bmatrix}\Lambda_G(n:n-m+1)\Lambda_H & 0\\ 0 & 0\end{bmatrix}\right)\\ &= \sum_{1\leq i\leq m}\lambda_i(W_H)\lambda_{n-i+1}(D_G),\end{aligned}$$

where $\Lambda_G(n:n-m+1) = diag(\lambda_n(D_G), \lambda_{n-1}(D_G),\ldots,\ \lambda_{n-m+1}(D_G))$, and $\Lambda_H = diag(\lambda_1(W_H), \lambda_2(W_H),\ldots,\lambda_m(W_H))$. The second equation follows from the fact that $\lambda_k(W_H) = 0$ as long as $m \leq k \leq n$. ■

Theorem 1 asserts that once the permutation matrix P is known by $P = [Q_G(n:n-m+1)Q_H^T,\ Q_G(n-m:1)]$, the virtual network mapping f satisfying the optimization formula (1) comes from formula:

$$\forall v_i \in V(H)(1 \leq i \leq m), f(v_i) = P[u_1,u_2,\ldots,u_n]^T[i], \quad (13)$$

where $[u_1,u_2,\ldots,u_n]^T(u_i \in V(G))$ is the vector of nodes in substrate network in the same order of D_G.

B. Alignment of Distance Matrix Eigenspaces

For the distance matrix D_G, even if the eigenpairs $((\lambda_i(D_G),\ q_i(D_G))(1 \leq i \leq n)$ and $((\lambda_i(W_H),\ q_i(W_H))$ $(1 \leq i \leq m)$ and the association matrix P has been resolved, the classical Hungarian method cannot be applied directly on P to discover an optimal mapping f with regard to the embedding cost metric, because this solution will not be acceptable once one node pair correspondence prescribed by f violates the node and link constraints. To guarantee that solutions derived from Theorem 1 meet the node and link constraints, the association matrix P should qualify the node correspondence with dependency not only on cost but on the minimum constraints. This requires that the eigenspaces of D_G and W_H be further aligned.

Ideally, if there is a positive real number θ such that $\lambda_i(D_G) = \theta\lambda_i(W_H)$ holds for all eigenvalues $\lambda_i(W_H)$ of W_H, we may claim that each $\lambda_i(W_H)$ has been aligned to $\lambda_i(D_G)$ completely. Under this circumstance, the VNE solution is guaranteed to not violate the link constraint. This ideal situation requires that eigenvalues of G be distributed evenly and densely in a real interval. In a majority of situations, as long as all distinct eigenvalues $\lambda_i(W_H)$ of W_H are aligned well enough to $\lambda_i(D_G)$, the link and node demands from a VNR can still be satisfied. A well enough alignment between two m-dimensional eigenvalues vectors $\lambda(W_H) = [\lambda_1(W_H),\ldots,\lambda_m(W_H)]$ and $\lambda(D_G) = [\lambda_{n-m}(D_G), \lambda_{n-m+1}(D_G),\ldots,\lambda_n(D_G)]$ means shifting the position of $\lambda_i(D_G)$ so as to make $\lambda(D_G)$ and $\lambda(W_H)$ as collinear as possible. The following analysis suggests that if P results in a complete or approximately complete alignment, $([P^TD_GP]_{1:m} - W_H)(i,j) \geq 0$ must hold, as long as $1 \leq i,j \leq m$. Consider that $(\lambda_i(D_G), q_i(D_G))$ and $(\lambda_i(W_H), q_i(W_H))$ are increasing eigenpairs, namely $\lambda_1(D_G) \leq \lambda_2(D_G) \leq \ldots \leq \lambda_n(D_G)$ and $\lambda_1(W_H) \leq$

$\lambda_2(W_H) \leq \ldots \leq \lambda_m(W_H)$. By the Perron-Frobenius theorem ([19, p. 167]), eigenvalues of a nonnegative matrix are subject to

$$|\lambda_i(D_G)| \leq \lambda_n(D_G) \text{ and } |\lambda_i(W_H)| \leq \lambda_m(W_H).$$

This enables alignment of two eigenvalue vectors to be performed in a normalized space as $-1 \leq \lambda_i(D_G)/\lambda_n(D_G) \leq 1$ and $-1 \leq \lambda_i(W_H)/\lambda_m(W_H) \leq 1$.

For alignment of two series of eigenvalues, we select one $\lambda_i(D_G)$ which is closest to $\lambda_i(W_H)$ as the correspondence for each of $\lambda_i(W_H)$. In order to guarantee that link constraints are satisfied, the selected $\lambda_i(D_G)$ should satisfy $\lambda_i(D_G) \geq \lambda_i(W_H)$ such that the scope of choosing $\lambda_i(D_G)$ can narrow to the interval $[\lambda_i(W_H), \lambda_{n-m+i}(D_G)]$. The existence of $\lambda_i(D_G)$ in $[\lambda_i(W_H), \lambda_{n-m+i}(D_G)]$ can be ensured by the application of the Cauchy eigenvalues interlacing theorem ([19, p. 269]) which asserts that each eigenvalue $\lambda_i(W_H)$ of a principal submatrix of a symmetric matrix W_H is bounded by eigenvalues $\lambda_i(D_G)$ of D_G through the inequality:

$$\lambda_i(D_G) \leq \lambda_i(W_H) \leq \lambda_{n-m+i}(D_G).$$

Hence, $\lambda_i(D_G)$ can now be located in interval $(\lambda_i(W_H)/\lambda_m(W_H), \lambda_{n-m+i}(D_G)/\lambda_m(W_H))$. Select $\lambda_i(D_G)$ and write it as:

$$\lambda_i^*(D_G) = \arg \min_{\lambda_i(D_G)} \{\lambda_i(D_G) - \theta\lambda_i(W_H)\}.$$

Finding such a perfect $\lambda_i^*(D_G)$ so that $\lambda_i(W_H) = \theta\lambda_i^*(D_G)$ for each $\lambda_i(W_H)$, implies that we can rearrange columns of Q_G such that those eigenvectors $q_i^*(D_G)$ of D_G corresponding to $\lambda_i^*(D_G)(n-m+1 \leq i \leq n)$ are located in columns 1 to *m*. Let $Q_G^*(m+1:n)$ denote the submatrix consisting of the remaining columns of Q_G, namely:

$$Q_G^*(1{:}m) = [q_1^*(D_G), q_2^*(D_G), \ldots, q_m^*(D_G)].$$

Then, letting $\Lambda_G^*(1:m) = diag(\lambda_1^*(D_G), \lambda_2^*(D_G), \ldots, \lambda_m^*(D_G))$ and configuring the associated matrix $P = [Q_G^*(1:m)Q_H^T, Q_G^*(m+1:n)]$ will lead to:

$$\left[P^T D_G P\right]_{1:m} = \begin{bmatrix} Q_H\Lambda_G^*(1:m)Q_H^T & 0 \\ 0 & \Lambda_G^*(m:n) \end{bmatrix}_{1:m} = Q_H\Lambda_G^*(1:m)Q_H^T.$$

Thus, if we let $R = ([P^T D_G P]_{1:m} - W_H)$, then $R = Q_H(\Lambda_G^*(1:m) - \Lambda_H)Q_H^T \approx Q_H(diag((\theta-1)\lambda(W_H))Q_H^T = (\theta-1)W_H$, which in turn leads to $R(u,v) \geq 0$. Perron-Frobenius theorem confirms that λ_n is bounded by the inequality

$$n \min_i a(u_i) = \min_i \Sigma_j D_G(u_i, u_j) \leq \lambda_n,$$

where $a(u_i) = (1/n)\Sigma_{u_j} \min\{c(l_g)|l_g \in E(p(u_i, u_j))\}$ denotes the average of all minimum constraints lying on the paths $p(u_i, u_j)$, as u_j ranges over all SN nodes. Generally speaking, we must have $a(u_i) > 0$ due to the connectivity of SN. This inequality guarantees that λ_n will be large enough as long as the scale of SN, reflected in the number of nodes and links, grows, which is a fundamental feature in the current Internet infrastructure.

C. Eigenvector Sign Correction

On the other hand, we note that sign correction of D_G can also affect the computation of the matrix *P*. The need for sign correction arises so as to make the eigenvectors $q_i(D_G)$ collinear to eigenvectors $q_i(W_H)$ to avoid ambiguity. The most straightforward way of doing this is to take the absolute values over all $q_i(D_G)$ components, an operation that is quite common in graph matching. However, we have experimentally observed that this approach is unsuitable to VNE goals, due, perhaps, to subtle differences in the two application domains. Thus, we borrow another sign correction strategy due to Park *et al.* [20], where each eigenvector $q_i(D_G)$ is compared with its counterpart $q_i(W_H)$ and the signs of eigenvector $q_i(D_G)$ will be corrected as long as:

$$||q_i(D_G) + q_i(W_H)|| < ||q_i(D_G) - q_i(W_H)||$$

This approach implies that more components of $q_i(D_G)$ have opposite signs, so that $q_i(D_G)$ and $q_i(W_H)$ form a relatively large angle in eigenspace. In this case, $q_i(D_G)$ is corrected as $-q_i(D_G)$, which is still a valid eigenvector of D_G.

D. Using Generalized Distance Matrices

We applied the all-pairs shortest path distance matrix D_G, marked as PD, as an initial step to address the VNE optimization formulation. It is quite natural now to try to generalize D_G to other matrix forms of distance metrics, including the Euclid distance matrix (ED), the heat kernel distance matrix (HD), the minimum constraint distance matrix (MD), based on the standard adjacency matrix A_G. The all-pairs shortest path distance matrix D_G offers a good match to the VNE problem, but computation of all-pairs shortest path distance from the standard adjacency matrix A_G, by means of the “funny” matrix multiplication (described in [24]), is not convenient. The Euclid distance matrix (ED) and the heat kernel distance matrix (HD) can be used to avoid the computation of PD. Our experiments illustrate both of these distance matrices produce results that are comparable with those of PD, but at a higher computational speed.

1) Euclid Distance Matrix (ED): Let *D*(*x*) denote a column vector of *D* indexed by node *x* of the associated graph. In most circumstances, distance matrix *D* described here can be approximated by its Euclidean distance in corresponding eigenspace using node coordinates under the eigenvector basis, namely, the column indexed by *u* in *D* can be represented as

$$D(u) = \sum_{1 \leq i \leq |V|} \lambda_i q_i(u) q_i$$

which implies that the column vector *D*(*u*) has coordinates $[\lambda_1 q_1(u), \lambda_2 q_2(u), \ldots, \lambda_{|V|} q_{|V|}(u)]^T$ under basis $\{q_i | 1 \leq i \leq |V|\}$. This produces the following Euclidean distance between nodes *u* and *v*:

$$d_E^2(u,v) = ||D(u) - D(v)||_2 = \sum_{1 \leq i \leq |V|} \lambda_i^2 (q_i(u) - q_i(v))^2$$

When implementing the eigendecomposition for D_G and W_H, the component signs in columns of Q_G and Q_H must have been corrected to avoid sign ambiguity.

2) Heat Kernel Distance Matrix (HD): One issue not yet considered might substantially impact the quality and efficiency of embedding: Namely, the consequences of embedding multiple dynamically arriving VNRs. Models and algorithms have yet to address this problem. In order to enable dynamical embedding, the natural inclination is to let the distance matrix of SN update the reachable path between all node pairs before embedding the next VNR. Unfortunately, recalculation of stationary distributed shortest paths among all node pairs of SN might not be acceptable for a majority of VNRs. Hence, we consider alternatives to the shortest distance matrix, namely, resorting to other distance measures that capture bandwidth and path-length information of the graph over time. A natural alternative to use is the heat kernel distance [21]. Heat kernel is widely known as the fundamental solution of the heat equation, where the continuous Laplace-Beltrami operator acts on a time-varying scalar function $h : M \times R^{+} \rightarrow R$ on the manifold *M* to formulate the rate of heat change with time by equation

$$\frac{\partial h(x,t)}{\partial t} = -\Delta h(x,t)(x \in M).$$

Let $L = I - \Delta^{-1/2} D \Delta^{-1/2}$ be Laplacian matrix associated with *D*, and ϕ_i be the eigenvectors of *L*, Δ being the diagonal matrix with (*u, u*) element defined by $\Delta(u,u) = \sum_{v \sim u} D(u,v)$. The impulse response solution $h_t(u,v)$ of the heat equation above can be derived by exponentiating the normalized graph Laplacian *L* of discretized manifold *M* as

$$h_t(u,v) = e^{-tL}(u,v) = \sum_{1 \le i \le |V|} exp(-\lambda_i t)\phi_i(u)\phi_i(v), \tag{14}$$

which specifies the rate of information flow across the edges (*u,v*).

To quantity the shortest path length across nodes *u* and *v* in SN, a suitable alternative is to embed heat measured on SN nodes to an *r*-dimensional vector space, where the dimension of nodes is reduced from *n* to *r* to enhance the efficiency at the required accuracy level. Let the embedded vector space have basis vector $\{\phi_i | \phi_i = [\phi_{1i}, \phi_{2i}, \ldots, \phi_{ri}]^T\}$. We can rewrite equation (14) in the form of the column vector $h_t(v)$, indexing the *v* node in terms of coordinates expression under a basis of embedded vector space, namely:

$$h_t(v) = \sum_{1 \le i \le |V|} exp(-\lambda_i t)\phi_i \phi_i(v).$$

The foregoing implies, as long as setting vectors $[\phi_1, \phi_2, \ldots, \phi_{|V|}]$ as a basis of Euclidean space embedded SN, that an SN node *v* corresponding to the column $h_t(v)$ has coordinates expression

$$h_t(v) = [exp(-\lambda_1 t)\phi_1(v), exp(-\lambda_2 t)\phi_2(v), \\ \times\ exp(-\lambda_{|V|} t)\phi_{|V|}(v)]^T.$$

Using coordinates expressions $h_t(u)$ and $h_t(v)$ of nodes *u* and *v*, the Euclidean distance $d_E(u,v)$ can be computed for alternating the adjacency distance,

$$d_E^2(u,v) = ||h_t(u) - h_t(v)||_2 \\ = \sum_{|1 \le i \le |V|} exp(-2\lambda_i t)(\phi_i(u) - \phi_i(v))^2.$$

such that the distances between nodes and their neighbor clouds of nodes can be associated with eigensystem alignment of virtual and substrate nodes. By integrating equation (14) with respect to *t*, the commute-time kernel, interpreted as the transition probability density between (*u,v*) of a random walk of arbitrary length can be obtained:

$$T(u,v) = \sum_{1 \le i} (1/\lambda_i)\phi_i(u)\phi_i(v).$$

At the same time, noting that $L = I - D$, we can extend equation (14) with MacLaurin formula:

$$h_t = e^{-t(I-D)} = e^{-t} \sum_{k \ge 1} (t^k/k!) D^k.$$

The distance matrix *D* has been normalized through $D^*(u,v) = 1/\sqrt{\delta(u)\delta(v)}$, where $\delta(u) = \sum_{v \sim u} D(u,v)$ is referred to as the weighted degree of node *u*.

3) The Minimum Constraint Distance Matrix (MCD): As mentioned previously, sometimes the association matrix *P* derived from PD, ED and HD fails to guarantee that solutions meet the node and link constraints which lead to an unsatisfactory acceptance ratio. To avoid this problem, the distance matrix D_G can be replaced by new minimum constraint matrix (MCD) with dependency not only on cost but on the minimum constraints, as we have done in our simulations.

The minimum constraint distance (MCD) matrix whose entry $D_G(u_g, v_g)$, representing the minimum link capacity of links on the shortest path between node u_g and v_g, serves to associate the constraints of nodes and links. The minimum constraint matrix is still denoted as D_G, without causing confusion, because it can be also understood as a generalized form of distance matrix between nodes. Let each virtual or substrate link hold a weight indicating its bandwidth and consider the constraints in two incident nodes, namely

If $u_g \neq v_g$ then $D_G(u_g, v_g) = \min\{c(l_g) | l_g \in E(p(u_g, v_g))\}$ else $D_G(u_g, v_g) = c(u_g)$.

E. A Family of VNE Algorithms Using Distance Matrices

1) DMEA-X (A Family of VNE Algorithms): The process of solving a generic VNE problem described above may be organized as a family of algorithmic solutions, called distance matrix eigenspace alignment DMEA-X, to be described next.

Initially, the algorithm DMEA-PD is presented with a form of matrix optimization using the unweighted all-pairs shortest path distance (PD). Next, two distance forms in graph eigenspace, the Euclid and heat kernel distance measures, are considered in algorithms DMEA-ED and DMEA-HD, for disposing the dynamically derived VNR. Algorithm DMEA-HD imitates the process of heat diffusion to estimate the usage of network resources in time slots for dealing with dynamic VNE. Finally, for taking link and node constraint into account, algorithm DMEA-MCD with higher accuracy than DMEA-2D

Algorithm 1 DMEA-X

```
Input: virtual network $H = (V(H),E(H))$ and substrate network $G = (V(G), E(G))$;
Output: virtual network mapping $f$ which embeds $V(H)$ and $E(H)$ onto $V(G)$ and $E(G)$, respectively;
1: create the distance matrix $D_G$ and the weighted adjacency matrix $W_H$ from previous formulation;
2: implement the eigendecomposition of $D_G$ and $W_H$ as $D_G = Q_G \Lambda_G Q_G^T$ and $W_H = Q_H \Lambda_H Q_H^T$;
3: project all rows of $D(G)$ onto its eigen subspace with dimension equal to the dimension of $W_H$ by using $m$ leading eigenvectors;
4: solve the optimization problem (10) to calculate the association matrix $P = [Q_G(n : n - m + 1) Q_H^T, Q_G(n - m : 1)]$;
5: compute the final mapping $f$ from $P$ by formula (13).
```

is proposed, where the (*u,v*) entry of the distance matrix is the minimum constraint of links on shortest path between node pair (*u,v*).

This class of algorithms, indicated by symbol DMEA-X with distance metric X chosen as PD, ED, HD and MCD, are explained in Table IV.

2) Computation of Unweighted Shortest Path Distance Matrix (PD): The "funny" matrix multiplication (described in [24]) generates D_G as $D_G = A_G \min_{.} + A_G$, where the operator $\min_{.}+$ calculates the minimum path by summing all entities in row *i* and ones in column *j* pairwise.

$$D_G(u, v) = \min_x \{A_G(u, x) + A_G(x, v)\}.$$

The 0 entries of standard adjacency matrix A_G should be initialized as a user-defined constant number that is greater than the path length value *k*, a predefined parameter used by shortest-path-type VNE algorithms. At the same time, nodes *x* lying on the path *p*(*u,v*) are recorded to trace links to prepare for the link mapping stage:

$$E(p(u, v)) = \arg\min_x \{A_G(u, x) + A_G(x, v)\}.$$

3) VNE Algorithm DMEA-2D (Using 2D Approximation of Distance Matrices): With regard to the computational efficiency of algorithm DMEA-X, although the eigendecomposition of the distance matrix D_G is polynomially solvable, it generally imposes an $O(n^3)$ computational overhead. Some researchers have discovered that the Perron-Frobenius eigenvector of a nonnegative matrix is particularly suitable for the node ranking problems in graphs.

This motivates us to devise an approximate algorithm called DMEA-2D for enhancing efficiency of VNE without a significant loss of embedding quality. In DMEA-2D, we can rank VN and SN nodes by the weighted node degree calculated from the distance matrix D_G. This ranking can be computed with a comparatively low overhead, because the eigenvector corresponding to the maximal eigenvalue of the distance matrix has a closed form solution that is the Perron vector. We thus propose an inexact but efficient algorithm called DMEA-2D by projecting rows of D_G into its two-dimensional eigenspace for realizing it.

Algorithm 2 DMEA-2D

```
Input: virtual network $H = (V(H),E(H))$ and substrate network $G = (V(G), E(G))$;
Output: virtual network mapping $f$ which embeds $V(H)$ and $E(H)$ onto $V(G)$ and $E(G)$, respectively;
1: Calculate the node degree vectors $\delta_1(G)$ and $\delta_1(H)$ (Perron vector) of matrices $D_G$ and $W_H$;
2: Calculate the eigenvectors $\delta_2(G)$ and $\delta_2(H)$ corresponding to the second largest eigenvalues of matrices $D_G$ and $W_H$;
3: Rank nodes of $G$ and $H$ by values $\delta_1(\bullet)$ and $\delta_2(\bullet)$, in descending order;
4: For each node $v_h$ in sorted VN node list, find the node $v_g$ in sorted SN node list that meets the demand of $v_h$, perform node mapping $f(v_h) = v_g$;
5: For each virtual link $l_h = (u_h, v_h) \in E(H)$, for each substrate link $l_g$ on the shortest path between $f(u_h)$ and $f(v_h)$, perform link mapping $f(l_h) = l_g$;
```

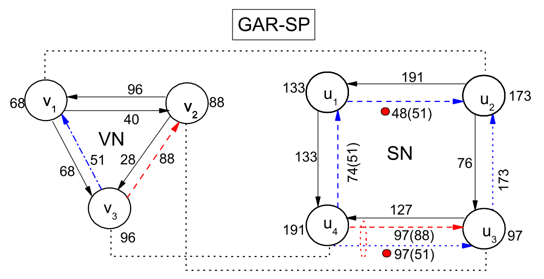
F. A Simple Example Illustrating DMEA-MCD

To gain intuition on challenges facing us in previous VNE approaches, we experiment with three representatives of existing VNE proposals, GAR-SP, RW-MM-SP and DViNE-SP, to implement embedding between simple virtual network C_3 and substrate network C_4 with the aid of the open-source framework for VNE simulation Alevin 2.2 [22]. Note that the network generator of Alevin 2.2 commonly generates nodes of the virtual network C_3 and substrate network C_4 as two ordered sequences $\{v_1, v_3, v_2\}$ and $\{u_1, u_4, u_2, u_3\}$, respectively, with node tags not in natural order.

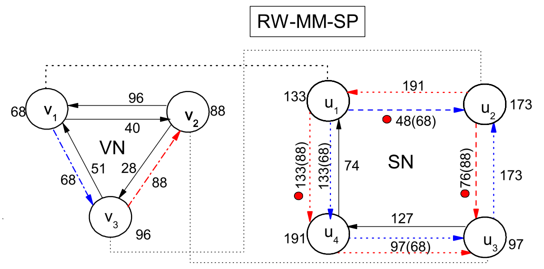
As discussed in Section I, the class of optimization VNE approaches usually search and discover the optimal solution globally. Although the DViNE-SP, as a representative of the type of VNE proposed in [3], obtains the optimal solution with near-optimal quality. Table VIII shows that DViNE-SP needs 1259.6s for embedding a VN of 58 nodes to a 100-node SN. Generally, this is unacceptable for practical VNE deployment. As significant alternatives, two other heuristic VNE solutions, GAR-SP [5] and RW-MM-SP [7], were proposed to perform VNE through approximate topological correspondence between VN and SN nodes. GAR-SP maps each VN node to an SN node with the largest residual resources.

To discover the node with maximal resource, this approach initializes the candidates of each VN node as identical ones consisting of all SN nodes. Strictly speaking, the global topological correspondence was taken into consideration to some extent. Consequently, a VNR will be rejected even if node mapping has been successful, for there may have been virtual link demand beyond constraints on embedded SN links.

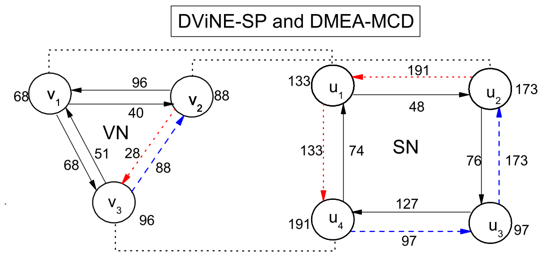
Low acceptance ratio attributed to uncoordinated node and link mapping constitutes the most essential challenge in our simulation verification. For observation of this point, Fig. 2a illustrates that GAR-SP gains the node mapping $\{v_1 \leftrightarrow u_2$, $v_2 \leftrightarrow u_3$, $v_3 \leftrightarrow u_4\}$ for embedding C_3 to C_4 after ordering the candidates of each VN node according to their available



(a) GAR-SP



(b) RW-MM-SP



(c) DViNE-SP and DMEA-MA

Fig. 2. Example for understanding the challenges faced in previous VNE schemes. Small red circles denote unsuccessful link mappings.

resource and choosing one having most resource. The candidates with resource indicators are calculated and expressed as $\{(u_1, 24073.0),\ (u_2,\ 46191.0)_,(u_3, 29100.0),\ (u_4, 32661.0)\}$. Then all virtual nodes in order $v_1 v_3 v_2$ greedily locate their targets as the sequence $u_2 u_4 u_3$, and this leads to the node mapping $\{v_1 \leftrightarrow u_2, v_2 \leftrightarrow u_{3,} v_3 \leftrightarrow u_4\}$, by associating the sequence $v_1 v_3 v_2$ with sequence $u_2 u_4 u_3$. Unfortunately, subsequent link mapping fails to map the link $v_3 \rightarrow v_1$ because $v_3 \rightarrow v_1$ is first mapped to the optimal target $u_4 \rightarrow u_1 \rightarrow u_2$, but this violates the constraint. Subsequently, the unique possible alternative $u_4 \rightarrow u_3 \rightarrow u_2$ also violates the constraint on link $u_4 \rightarrow u_3$, since the link $v_3 \rightarrow v_2$ has already occupied a bandwidth 88 prior to $v_3 \rightarrow v_1$.

RW-MM-SP formulates an iterative process over VN and SN to converge a random walk to the stationary status, where *r*(*G*) and *r*(*H*), two decreasingly sorted real-value sequences, are derived iteratively, and nodes are ranked according to their corresponding values in sequences *r*(*G*) or *r*(*H*).

$r(G) = \{(u_2, 0.0014)_,(u_3, 0.0013), (u_1, 0.0011),$
$(u_4, 0.0010)\};$
$r(H) = \{(v_3, 0.0012)_,(v_2, 0.0011), (v_1, 0.0009)\}.$

Correspondence of *r*(*H*) with *r*(*G*) generates the node mapping $\{v_1 \leftrightarrow u_1, v_2 \leftrightarrow u_{3,} v_3 \leftrightarrow u_2\}$. Likewise, RW-MM-SP firstly attempts to map link $v_1 \rightarrow v_3$ to $u_1 \rightarrow u_2$, and $v_3 \rightarrow v_2$ to $u_2 \rightarrow u_3$ by the rule of shortest path first. However, bandwidths on both mapped links do not fulfill the virtual link demands. The algorithm switches the target of mapping link $v_1 \rightarrow v_3$ to the path $u_1 \rightarrow u_4 \rightarrow u_3 \rightarrow u_2$, and the one of $v_3 \rightarrow v_2$ to the path $u_2 \rightarrow \underline{u_1 \rightarrow u_4} \rightarrow u_3$. The first choice, mapping $v_1 \rightarrow v_3$ to path $\underline{u_1 \rightarrow u_4} \rightarrow u_3 \rightarrow u_2$, is successful thus it occupies a bandwidth 68 of link $u_1 \rightarrow u_4$, but this causes that latter link mapping from $v_3 \rightarrow v_2$ to the path $u_2 \rightarrow \underline{u_1 \rightarrow u_4} \rightarrow u_3$ to violate the bandwidth constraint on link $u_1 \rightarrow u_4$ (see the small red circle mark) for its residual bandwidth of 65(133−68) is less than the demand 88 of link $v_3 \rightarrow v_2$. Finally, embedding of C_3 to C_4 is unsuccessful by RW-MM-SP.

The fundamental reason, in part from conjecture but partly validated by our experiments, behind both GAR-SP and RW-MM-SP failing to embed C_3 to C_4 is in the ways they consider the topological correlation between virtual and substrate node. The former takes this factor into account through resource level of nodes and links connected it, and the latter formulates it as weights of nodes calculated from a candidate node and its neighbors. Both are confined to local characteristic correlation, which rarely utilizes coordination between nodes and links, leading to failure of link mapping following successful node mappings.

As a successful mapping example, Fig. 2c shows the result of DViNE-SP, which searches for the optimal candidate for each VN node, and also considers the correlation between the links connected to nodes for optimization. The node mapping $\{v_1 \leftrightarrow u_1, v_2 \leftrightarrow u_{2,} v_3 \leftrightarrow u_4\}$, which is the optimal one, follows from choosing the maximal weights from candidates

$v_1 \rightarrow \max\{(u_1, 0.9896), (u_2, 0.0030), (u_3, 0.0074)\};$
$v_3 \rightarrow \max\{(u_1, 0.7129), (u_4, 0.2359), (u_2, 0.0511), (u_3, 0)\};$
$v_2 \rightarrow \max\{(u_1, 0.0004), (u_4, 0), (u_2, 0.0.9972),$
$(u_3, 0.0024)\};$

As already mentioned, DViNE-SP suffers from its unacceptable runtime when the network scales up (sizes greater than 58 for VN and 100 for SN).

The motivation behind our work is to pursue trade-off between efficiency and precision. The results of embedding C_3 to C_4 demonstrate our success in reaching the near optimal solution, while also overcoming the computational intractability, so that runtime approaches to that of approximate matrix eigendecomposition. Regarding the example above, our scheme aligns the eigenspace of the minimum constraint distance matrix D_G for contributing a VNE solution of embedding C_3 to C_4 with identical quality to DViNE-SP, as depicted in Fig. 2c, but significant acceleration as seen in Table III, and further verified in subsequent simulation.

The overall process is illustrated as follows. For the distance matrix D_G of C_4, the matrix rows are indexed as the node order, the matrix rows being indexed as the node order $\{u_1, u_{4,} u_2, u_3\}$,and D_G has the form:

$$D_G = \begin{bmatrix} 133 & 133 & 48 & 48 \\ 74 & 191 & 48 & 97 \\ 191 & 76 & 173 & 76 \\ 74 & 127 & 173 & 97 \end{bmatrix};$$

TABLE III
EXAMPLE DETAILS FOR VNE FROM C_3 TO C_4

Algorithm	Runtime	Acceptance	Node Mapping
GAR-SP	0.014s	rejected	$\{v_1 \leftrightarrow u_2, v_2 \leftrightarrow u_3, v_3 \leftrightarrow u_4\}$
DViNE-SP	0.089s	accepted	$\{v_1 \leftrightarrow u_1, v_2 \leftrightarrow u_2, v_3 \leftrightarrow u_4\}$
RW-MM-SP	0.007s	rejected	$\{v_1 \leftrightarrow u_1, v_2 \leftrightarrow u_3, v_3 \leftrightarrow u_2\}$
DMEA-MCD	0.073s	accepted	$\{v_1 \leftrightarrow u_1, v_2 \leftrightarrow u_2, v_3 \leftrightarrow u_4\}$

Its eigenvalues matrix Λ_G and eigenvectors matrix Q_G are

$$Q_G = \begin{bmatrix} -0.38 & 0.48 & 0.35 & 0.21 \\ -0.46 & -0.12 & 0.56 & -0.44 \\ -0.58 & 0.09 & -0.74 & -0.43 \\ -0.56 & -0.62 & -0.55 & 1.00 \end{bmatrix};$$

and $\Lambda_G = diag(432.56, 89.60, 89.60, -17.77)$.

For the weighted adjacency matrix W_H of C_3, the rows of matrix are indexed as the node order $\{v_1, v_3, v_2\}$. The matrix W_H, its eigenvectors matrix Q_H and eigenvalues matrix Λ_H are

$$W_H = \begin{bmatrix} 68 & 68 & 40 \\ 51 & 96 & 88 \\ 96 & 28 & 88 \end{bmatrix}, Q_H = \begin{bmatrix} -0.49 & 0.45 & 0.42 \\ -0.67 & -0.51 & 0.43 \\ -0.56 & -0.01 & -0.80 \\ & & \end{bmatrix},$$

and $\Lambda_H = diag(206.34, 22.83, 22.83)$, respectively.

Through the procedure of sign correction, the signs of all Q_G columns remain unchanged. Thus columns *1* to *m* are extracted as a submatrix $Q_G(1:m)$ whose columns constitute the basis vectors of *m*-dimensional subspace that all C_4 nodes should be projected into

$$Q_G(1:m) = \begin{bmatrix} -0.38 & 0.48 & 0.35 \\ -0.46 & -0.12 & 0.56 \\ -0.58 & 0.09 & -0.74 \\ -0.56 & -0.62 & -0.55 \end{bmatrix};$$

relaxing the permutation matrix *P* as an orthogonal matrix leads to a generalized permutation matrix *P* with its p_{ij} component indexed by VN node *i* and SN node *j*. The nonnegative real number p_{ij} reflects the weight or possibility of embedding VN node *i* to SN node *j*. Regarding the optimization objective, we choose the SN node *j* maximizing p_{ij} in row *i* as the best candidate of VN node *i*. If this SN node has been previously chosen by another VN node, the second largest p_{ij} may dictate the target of mapping virtual node *i*, and so on. Finally, we obtain *P* as

$$P = \begin{bmatrix} 0.55 & 0.41 & 0.02 & 0.24 \\ 0.16 & 0.60 & 0.02 & 0.45 \\ 0.07 & 0.19 & 0.92 & 0.76 \end{bmatrix}.$$

For this example, *P* yields the node mapping $\{v_1 \leftrightarrow u_1, v_2 \leftrightarrow u_2, v_3 \leftrightarrow u_4\}$ between C_3 to C_4, the same as DViNE-SP's result.

IV. EXPERIMENTAL RESULTS

In this section, we experimentally verify the efficiency and quality of our scheme in terms of various VNE metrics, involving runtime, VNR acceptance ratio (AR), cost/revenue ratio (C/R), and node utilization ratio (NUR). The experimental platform uses the software IDE Eclipse Neon under the 32-bit Windows 7 operating system, and hardware CPU Intel Core i7 5600-U @2.6 GHz with 8.0 GB RAM. All simulations generate the results under the VNE simulation environment Alevin with recently updated software package version 2.2, developed by Fischer *et al.* [1], that has been recognized as a prime simulation framework for examining virtual network embedding algorithms.

TABLE IV
A FAMILY OF DISTANCE MATRICES FOR EIGENSPACE ALIGNMENT

Algorithm	**Characteristics**
DMEA-2D	Project SN and VN node to the 2 dimensional (2D) subspace spanning by two leading eigenvectors of the distance matrix.
DMEA-PD	using the unweighted shortest path distance (PD) matrix
DMEA-ED	using the Euclid distance (ED) of graph eigenspace
DMEA-HD	using the heat kernel distance (HD) of graph Laplace matrix
DMEA-MCD	using the minimum constraint distance matrix of weighted adjacency

We encoded algorithms based on graph eigensystem using Java programming language to implement all simulations under multiple experimental settings. We investigated three schemes based on eigenspace to demonstrate their performance and accuracy, measured by runtime and acceptance ratio, respectively. Five variants of our VNE scheme of aligning eigenspace, called DMEA-2D, DMEA-PD, DMEA-ED, DMEA-HD, DMEA-MCD, were analyzed and compared, with different behaviors due to aligning nodes of SN and VN with distinct distance metrics. Table IV shows the relevant details.

The process of each experiment can be decomposed into network generation, algorithm configuration and execution, and algorithm evaluation, with various experimental configurations. The experimental steps and corresponding configurations are further detailed in Table V. Next, the family of eigensystem alignment VNE approaches are compared with a couple of representative VNE algorithms, that have been cited as the focus of considerable VNE research, in terms of runtime, VNR acceptance ratio, cost/revenue ratio, and node utilization ratio, factors that have been recognized as effective means of assessing VNE algorithms. Finally, the results of comparison have been plotted to reveal the effect of varied scale of VN and SN on VNE performance and quality.

A. Scenario Generation

The step of network generation goes through establishing network topology, adding resources to SN, as well as adding demands to VN. The SN is modeled as random network, a property recognized as a significant characteristic of the Internet as it has evolved. Random topologies are generated with a probability 0.5 of connecting a pair of nodes. The numbers *n* and *m* of SN and VN nodes increase linearly with the iteration *k* of experiment $n = a(k+1)$ and $m = b(k+1)$, where *a* and *b* are constant factors controlling the growth of SN and VN, respectively. For configurations here, they hold values $a = 10$ and $b = 5$. The configuration parameters of network generation are listed in Table V.

TABLE V
EXPERIMENTAL PARAMETERS OF NETWORK GENERATION, a AND b BEING CONSTANT FACTORS

Value	SN Size	VN Size	VNs Num.	Demand	Resource
Min	10	5	5	10	10
Max	100	58	5	1000	1000
k-term	$a(k+1)$	$b(k+1)$	5	[10,1000]	[10,1000]

TABLE VI
REPRESENTATIVE VNE APPROACHES CHOSEN FOR EVALUATION, THE CITATION TIMES UPDATED ON JULY/08/2017

Algorithm	Reference	Brief description
DViNE-SP	Chowdhury et al. [3]	VNE with coordinated strategy in two stages where node mapping is implemented by mixed integer programming (MIP) and link mapping with k-shortest paths. Google Scholar [23] citations: 415
GAR-SP	Yu et al. [5]	VNE preferentially using available resources for node mapping and k-shortest paths for link mapping. Google Scholar [23] citations: 998
RW-MM-SP	Cheng et al. [7]	VNE ranking nodes with topology properties for node mapping and k-shortest paths for link mapping. Google Scholar [23] citations: 346

TABLE VII
COMPARISON OF THE AVERAGE VALUES OF THE EVALUATION METRICS FROM 10 TEST ITERATIONS EXECUTED ON RANDOMLY GENERATED VN AND SN

Algorithm	Runtime	Acceptance	C/R	NU
GAR-SP	1.1987	98%	1.3807	20.7342
DViNE-SP	134.9734	100%	1.4521	21.4261
RW-MM-SP	2.7386	96%	1.3485	21.8244
DMEA-2D	1.1925	100%	1.2868	23.6131
DMEA-ED	1.1454	100%	1.3737	20.4356
DMEA-HD	1.1975	100%	1.3680	20.5497
DMEA-PD	1.1038	100%	1.3961	20.3806
DMEA-MCD	1.1682	100%	1.3799	20.8616

B. Algorithm Configuration

Algorithms chosen for our experimental evaluation involve three representative VNE algorithms, which have been proposed in [3], [5], and [7], and briefly described in Table VI. These algorithms along with our proposed family of VNE eigensystem aligning approaches are executed on identical scenarios and parameter configurations. All algorithms chosen for evaluation were run 10 times under identical scenario set-ups, but increasing size of SN and VN, as shown in Table V. In the node mapping stage, CPU node weights are set to 1, and the candidates of a VN node are limited within a distance of 20 hops away from it. The situation of node overload has not yet been considered. In the link mapping stage, the parameter k of mapping a VN link to a length-k shortest path is set to $k = 2$, and the weights on link bandwidth conform to the uniform distribution in interval $[\min, \max]$ (see Table V).

C. Evaluation Results

The simulation proceeds with varying the network scale with time (see Table V). Our evaluation focuses on a collection of well-recognized VNE metrics (runtime, VNR acceptance ratio, cost/revenue ratio, and node utilization ratio) for assessing the competitiveness of our proposal. All variants of eigensystem aligning are expected to perform better than counterparts, at least with respect to runtime, without degradation in other metrics. We are particularly interested in efficiency of DMEA-2D and in trade-off of efficiency and embedding quality of DMEA-X. The results of performing all algorithms confirm that our strategy yields the expected runtime improvement. Likewise, it generates a high-quality approximate solution in terms of other evaluation metrics. This means that eigensystem-based algorithms improve efficiency, without a cost in quality degradation. Because we are primarily interested shortest-path class VNE approaches, experiments with various kinds of path splitting have not yet been performed.

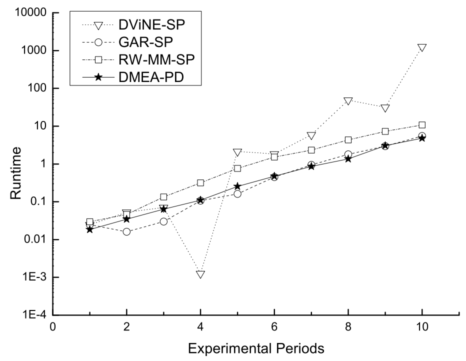
The results of comparisons with other algorithms in multiple metrics listed above are depicted in Figs. 3–4. We address the evaluation of DMEA-X in three domains: embedding efficiency, embedding quality, and scalability.

Metric 1 (Embedding Efficiency): Our simulation indicates that the optimization-based scheme DViNE-SP encounters efficiency problems when a task of embedding randomly generated SN with 100 nodes to VN with 58 nodes is attempted in the 10th iteration on random graphs, as shown in Fig. 3a. The problem is due to timeout caused by the optimization package GLPK 4.7 in searching for solutions.

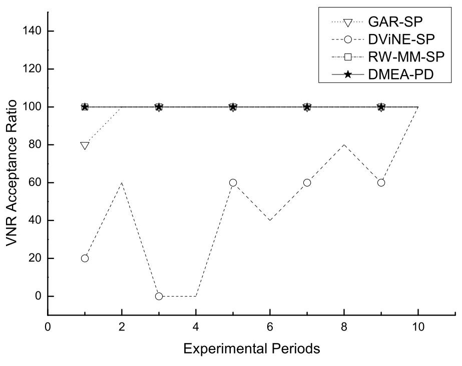
With respect to runtime, eigenspace-based schemes exhibit improvement over DViNE-SP and RW-MM-SP, and come extremely close to GAR-SP, with DMEA-PD(see Fig. 3a) and DMEA-2D (see Fig. 4a). The eigenspace alignment algorithms using other distance metrics demonstrate runtimes close to DMEA-PD but more than DMEA-2D (see Fig. 5a). The experiments leading to Figs. 3–5 were conducted on a pair of randomly generated VN and SN. Through a 10-iteration test, DMEA-X embedded 100 percent of virtual network requests (VNR acceptance ratio) within nearly equal average runtime 1.1s, which is less than GAR-SP's 1.1987. Notably, the other two tested algorithms, DViNE-SP and RW-MM-SP, exhibited poor efficiency with average runtimes of 134.97 and 2.74, as seen in Fig. 3a and Fig. 4a.

Metric 2 (Embedding Quality): The acceptance ratio, defined as the ratio between the number of accepted VNRs and the total number of VNRs, reflects the fraction of VNRs successfully embedded as virtual networks. The cost/revenue ratio, 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 cost/revenue ratio measures the proportion of cost spent in the substrate network, taking into account the revenue that has been mapped, the lower the cost/revenue ratio, 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.

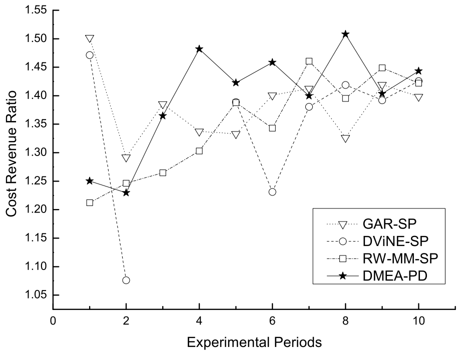
On randomly generated SNs, DMEA-X generates an average VNR acceptance ratio 100%, average cost/revenue ratio around 1.36, and the node utilization ratio around 21, slightly lower than those of DViNE-SP but rather close to the other two algorithms, as seen in Table VII. In more details, Figs. 3 and 4



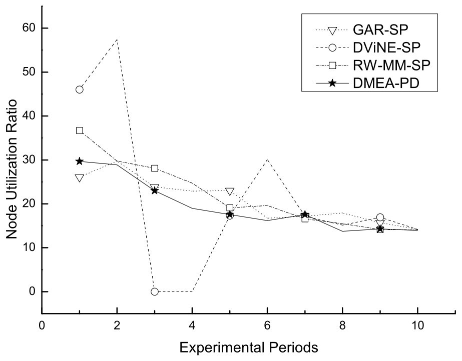
(a) Runtime (on log scale)



(b) VNR acceptance ratio

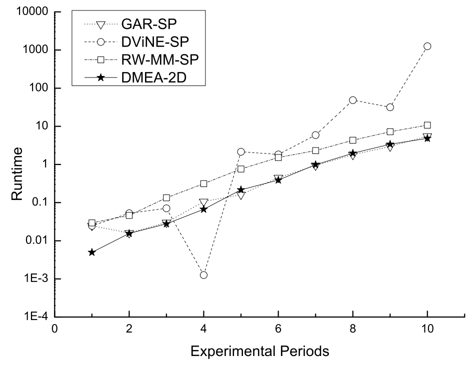


(c) Cost/revenue ratio

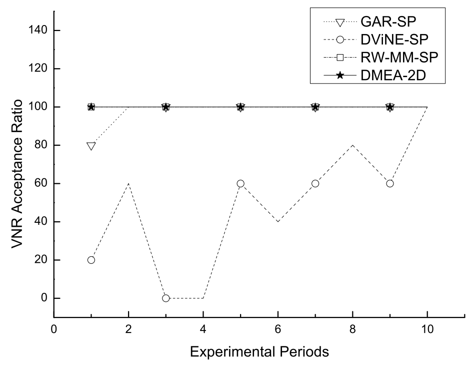


(d) Node utilization ratio

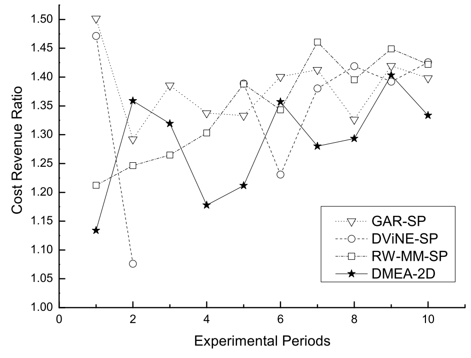
Fig. 3. Performance and quality comparisons of DMEA-PD with three representative VNE schemes for embedding a random VN into a random SN.



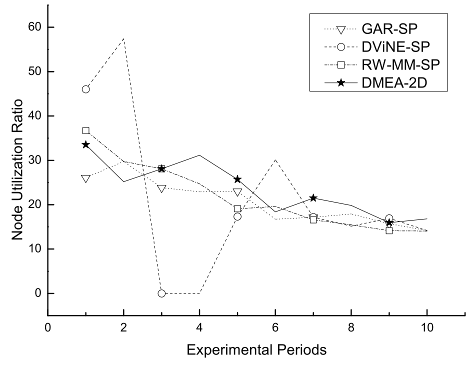
(a) Runtime (on log scale)



(b) VNR acceptance ratio



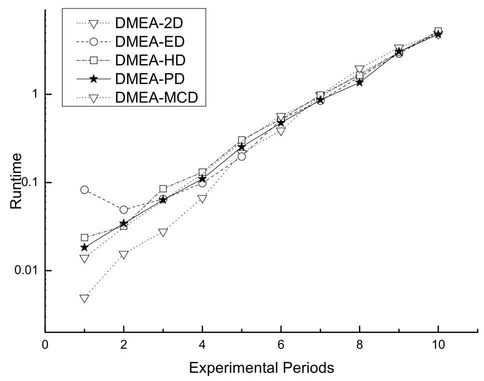
(c) Cost/revenue ratio



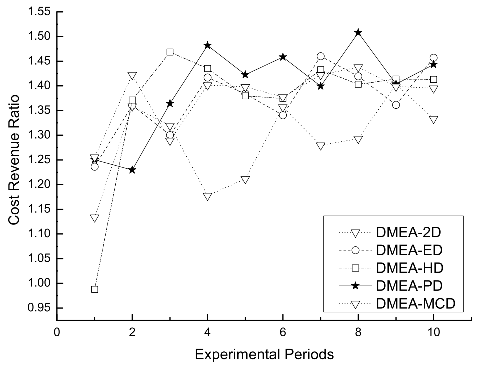
(d) Node utilization ratio

Fig. 4. Comparisons in performance and quality of DMEA-2D with three representative VNE schemes for embedding a random VN into a random SN.

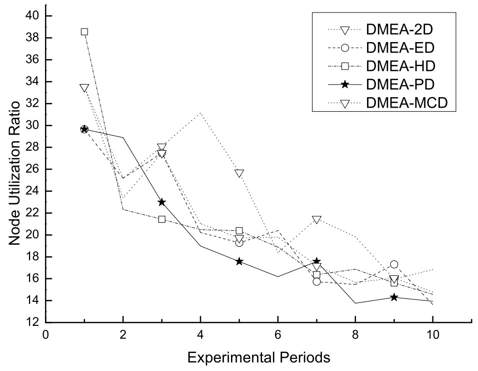
demonstrate that DMEA-X produces more competitive values with regard to the runtime metric than the other three approaches, while maintaining the expected quality. On the other hand, we also notice that sometimes the experimental results yield low node utilization ratio around 20%. This issue is also s problem in the other three compared schemes. The main reason for the lower node utilization ratio is that the node



(a) Runtime (on log scale)



(b) Cost/revenue ratio



(c) Node utilization ratio

Fig. 5. Comparisons in performance and quality of approaches with five distinct distance matrices for embedding a random VN into a random SN.

and link constraints for VN are configured as lower value relative to those of SN. Overall, our scheme is more capable of trading off performance and quality than other approaches.

Metric 3 (Scalability): In order to evaluate the scalability of our proposal, we experiment with increasing sizes of VN and SN, with a more extensive VN range, varying from 5 to 58. It can be observed from Figs. 3 and 4 (also from Table VIII) that the advantages of our approach grow for larger network sizes, that is, it exhibits a lower runtime growth with increasing VN and SN sizes. In the 10th iteration of our experiment, when the VN and SN networks reach the maximal sizes of 100 and 58, DMEA-X's runtime is around 5, compared with around 5.5, 10.7, and 1260 for its counterparts GAR-SP, RW-MM-SP, and DViNE-SP, respectively, with a nearly equal mapping quality.

TABLE VIII
A COMPARISON IN PERFORMANCE AND QUALITY OF VNE ALGORITHMS WHEN THE VN AND SN NETWORKS ARE ENLARGED TO MAXIMAL SIZES OF 100 AND 58 (REACHED IN THE 10TH ITERATION OF OUR EXPERIMENT), RESPECTIVELY, TO DEMONSTRATE THE SCALABILITY OF OUR STRATEGY

Algorithm	Runtime	Acceptance	C/R	NU
GAR-SP	5.5073	100%	1.3984	14.1883
DViNE-SP	1259.6000	100%	1.4256	14.1883
RW-MM-SP	10.7190	100%	1.4222	14.0622
DMEA-2D	4.8531	100%	1.3333	16.8298
DMEA-ED	5.1177	100%	1.4571	13.6379
DMEA-HD	5.2387	100%	1.4131	14.5806
DMEA-PD	4.8080	100%	1.4432	13.9383
DMEA-MCD	4.9380	100%	1.3951	14.7850

As stated earlier, achieving optimal or nearly optimal solution for addressing a general VNE problem is extremely challenging. In fact, a higher VNR acceptance ratio only reflects to what extent a VNR can be technically fulfilled. A holistic concept termed "acceptance" may be required to gauge the extent to which users and providers achieve an agreement, regardless of technical or business factors. In this regard, on the precondition of nearly equal quality, efficiency plays a most essential role to affect user's decision in the current user-centered environment.

V. CONCLUSION

Virtual network embedding is a fairly substantial component in network virtualization technology, and the latter has been identified as a promising approach to overcoming the ossification problem arising in future network architectures. The computational intractability of optimization-based approaches and the uncertain quality of heuristic algorithms have triggered vast amounts of research.

Our work facilitates the process of virtual network embedding with graph eigenspace alignment. We have conducted a theoretical analysis and a series of simulation studies on multiple scenarios to validate that our scheme based on graph eigenspace alignment can shorten the algorithm's runtime, while maintaining high quality for a general VNE task.

An important observation arising from our experimental work is that our scheme does not lead to a radically improved node and link utilization ratios. Theoretically, we are also interested in exploring hierarchical network topologies to handle VNE tasks in extremely large-scale networks, both VN and SN. Our future research will cover these two problems.

REFERENCES

[1] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 4th Quart., 2013.
[2] A. Jarray and A. Karmouch, "Decomposition approaches for virtual network embedding with one-shot node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 1012–1025, Jun. 2015.
[3] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, Feb. 2012.
[4] X. Chen, C. Li, and Y. Jiang, "Optimization model and algorithm for energy efficient virtual node embedding," *IEEE Commun. Lett.*, vol. 19, no. 8, pp. 1327–1330, Aug. 2015.

[5] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17–29, 2008.

[6] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proc. 1st ACM Workshop Virtualized Infrastruct. Syst. Archit.*, Barcelona, Spain, 2009, pp. 81–88.

[7] X. Cheng *et al.*, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 38–47, 2011.

[8] W. Fan, J. Li, S. Ma, H. Wang, and Y. Wu, "Graph homomorphism revisited for graph matching," in *Proc. VLDB Endow.*, vol. 3, 2010, pp. 1161–1172.

[9] H. Cui, S. Tang, X. Huang, J. Chen, and Y. Liu, "A novel method of virtual network embedding based on topology convergence-degree," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC)*, Budapest, Hungary, 2013, pp. 246–250.

[10] D. Zhang and L. Gao, "Virtual network mapping through locality-aware topological potential and influence node ranking," *Chin. J. Electron.*, vol. 23, no. 1, pp. 61–64, 2014.

[11] Y. Cao, W. Fan, and S. Ma, "Virtual network mapping: A graph pattern matching approach," in *Proc. Brit. Int. Conf. Databases*, 2015, pp. 49–61.

[12] F. Esposito, D. Di Paola, and I. Matta, "On distributed virtual network embedding with guarantees," *IEEE/ACM Trans. Netw.*, vol. 24, no. 1, pp. 569–582, Feb. 2016.

[13] M. T. Beck, A. Fischer, J. F. Botero, C. Linnhoff-Popien, and H. D. Meer, "Distributed and scalable embedding of virtual networks," *J. Netw. Comput. Appl.*, vol. 56, pp. 124–136, Oct. 2015.

[14] Z. Zhang *et al.*, "Adaptive multi-objective artificial immune system based virtual network embedding," *J. Netw. Comput. Appl.*, vol. 53, pp. 140–155, Jul. 2015.

[15] S. Zhang, Z. Qian, J. Wu, S. Lu, and L. Epstein, "Virtual network embedding with opportunistic resource sharing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 816–827, Mar. 2014.

[16] D. Knossow, A. Sharma, D. Mateus, and R. Horaud, "Inexact matching of large and sparse graphs using Laplacian eigenvectors," in *Proc. GbRPR*, vol. 9, 2009, pp. 144–153.

[17] R. Singh, J. Xu, and B. Berger, "Pairwise global alignment of protein interaction networks by matching neighborhood topology," in *Research in Computational Molecular Biology*. Berlin, Germany: Springer, 2007, pp. 16–31.

[18] R. Bhatia, *Matrix Analysis*, vol. 169. New York, NY, USA: Springer, 2013.

[19] F. Zhang, *Matrix Theory: Basic Results and Techniques*. New York, NY, USA: Springer, 2011.

[20] S. H. Park, K. M. Lee, and S. U. Lee, "A line feature matching technique based on an eigenvector approach," *Comput. Vis. Image Understanding*, vol. 77, no. 3, pp. 263–283, 2000.

[21] X. Bai and E. R. Hancock, "Heat kernels, manifolds and graph embedding," in *Proc. Struct. Syntactic Stat. Pattern Recognit. Joint IAPR Int. Workshops (SSPR SPR)*, Lisbon, Portugal, Aug. 2004, pp. 198–206.

[22] M. T. Beck, C. Linnhoff-Popien, A. Fischer, F. Kokot, and H. De Meer, "A simulation framework for virtual network embedding algorithms," in *Proc. 16th Int. Telecommun. Netw. Strategy Plan. Symp. (Netw.)*, 2014, pp. 1–6.

[23] *Google*. Accessed: Jul. 8, 2017. [Online]. Available: https://scholar.google.com

Chenggui Zhao received the bachelor's degree in mathematics and the master's degree in computer science from Yunnan Normal University, China, in 1999 and 2003, respectively, and the Ph.D. degree in computer science from the South China University of Technology, China, in 2007. He was a Visiting Academic Researcher with the Department of Electrical and Computer Engineering, University of California at Santa Barbara from 2016 to 2017. He is currently a Full Professor with the School of Information, Yunnan University of Finance and Economics, China. His research focuses mainly on communication network and computer system architecture, studied via the application of algebra, graph, and probability theories. He is also involved in the areas of data and social network analysis.

Behrooz Parhami (M'76–SM'81–F'97–LF'13) is a Professor with the Department of Electrical and Computer Engineering, and the Former Associate Dean for Academic Personnel with the College of Engineering, University of California, Santa Barbara, where he teaches and studies computer hardware architecture. He has written six textbooks and over 300 peer-reviewed technical papers. He was a recipient of several other awards, including a Most Cited Paper Award from the *Journal of Parallel and Distributed Computing*. He serves on journal editorial boards (currently serving on two IEEE Transactions) and conference program committees and is also active in technical consulting. He is a life fellow of IEEE, a fellow of IET and British Computer Society.