entropy

MDPI

Article

Virtual Network Embedding Based on Graph Entropy

Jingjing Zhang [1,†], Chenggui Zhao [1,*,†], Honggang Wu [2], Minghui Lin [1] and Ren Duan [1]

[1] School of Information, Yunnan University of Finance and Economics, Kunming 650221, China; zhangjj0710@gmail.com (J.Z.); lmh@ynufe.edu.cn (M.L.); duanren@ynufe.edu.cn (R.D.)
[2] School of Continuing Education, Yunnan University of Finance and Economics, Kunming 650221, China; wuhg@ynufe.edu.cn
* Correspondence: zhaochenggui@126.com; Tel.: +86-871-6511-3839
† These authors contributed equally to this work.

Received: 25 January 2018; Accepted: 21 April 2018; Published: 25 April 2018

Abstract: For embedding virtual networks into a large scale substrate network, a massive amount of time is needed to search the resource space even if the scale of the virtual network is small. The complexity of searching the candidate resource will be reduced if candidates in substrate network can be located in a group of particularly matched areas, in which the resource distribution and communication structure of the substrate network exhibit a maximal similarity with the objective virtual network. This work proposes to discover the optimally suitable resource in a substrate network corresponding to the objective virtual network through comparison of their graph entropies. Aiming for this, the substrate network is divided into substructures referring to the importance of nodes in it, and the entropies of these substructures are calculated. The virtual network will be embedded preferentially into the substructure with the closest entropy if the substrate resource satisfies the demand of the virtual network. The experimental results validate that the efficiency of virtual network embedding can be improved through our proposal. Simultaneously, the quality of embedding has been guaranteed without significant degradation.

Keywords: graph entropy; virtual network embedding; probability; information measure

MSC: 68M10

1. Introduction

1.1. Virtual Network Embedding

Network virtualization (NV) technology enables flexibility [1] on relatively rigid Internet architecture to accommodate gradually abundant Internet applications. Successful paradigms have emerged in cloud data centers for resource allocation [1], which stimulates a considerable volume of work for efficient solutions in this field. For realizing network virtualization, Internet Service Provider (ISP) must provide a mechanism for allocating substrate physical resources to provide user-expected services by ISPs. The resource allocation function in NV is presented as virtual network embedding (VNE), here "embedding" is also equivalently termed mapping, provisioning or assignment. Generally, VNE formulates user demand as virtual networks (VNs) consisting of virtual nodes connected by virtual links, and substrate physical resources as substrate network (SN).

Research on the VNE problem originated from finding the optimal solution in respect to some object to configure distributed substrate resource for fulfilling the user's request. Theoretically, this requires mathematical optimization to formulate and solve the VNE problem accurately. Chowdhury et al. [2] extend the substrate network to an augmented substrate graph in which virtual nodes are connected to substrate nodes within some distance so as to merge both into a meta graph. Over the meta graph,

the VNE problem is formulated by a mixed-integer program, and two approaches, deterministic D-ViNE and randomized R-ViNE, are devised for solving. Jarray et al. [3] decompose an overall VNE problem into a main part with constraints on the availability of substrate resource, and a pricing part with constraints on the embedding of VN requests, where competition among VN users is concerned through a periodical auction, and problem formulation is facilitated with column generation technology (CG). Besides parameters such as provider revenue, acceptance ratio, and embedding cost (see, e.g., [1]), Chen et al. [4] construct a minimization model of energy-efficient virtual node embedding and solve it to yield the minimal product of energy.

In parallel, heuristic solutions for the VNE problem receive much attention because researchers try to overcome the low implementation efficiency on mathematical optimization model. Yu et al. [5] advocate maximizing 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. Then virtual links are mapped to the shortest paths between two mapped substrate nodes. Lischka et al. [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. The VNE algorithm through isomorphism detection performs particularly efficiently, compared to other two-stage approaches on large VNs with high resource demands. Cheng et al. [7] propose a VNE strategy based on node ranking. Their approach ranks all virtual and substrate nodes according to their relative importance measured by the bandwidths on outgoing links, and the importance of all reachable nodes and out-neighbors. Then the higher-ranked virtual nodes have priority to be mapped to the higher-ranked substrate nodes. Virtual link mapping is implemented by the shortest-path algorithm or by the multi-commodity flow algorithm.

Recently, Beck et al. [8] design a distributed and parallel VNE framework called DPVNE, in which several VNE algorithms are executed in a distributive way, so that the single point pressure in SN is alleviated and greater efficiency is achieved. Zhang et al. [9] propose an opportunistic resource-sharing scheme to handle time-dependent virtual network requests (VNRs), in which the variable section is separated from required resources, and two solutions of allocating time slot are proposed such that bandwidth of virtual links can be mapped in corresponding time slots for realizing multiple VNs sharing substrate resources meanwhile.

Unfortunately, the VNE approaches introduced so far commonly encounter a challenge that they will consume unacceptable time when executing combinatorial search for solutions in large objective spaces. Naturally, we consider relying on graph entropy of the substructure for time complexity reduction. Such motivation arises from the fact that graph entropy can capture the randomness of the local substructure, which can confine the candidate objects of a demanded resource to a set of subnetworks having enough available resources with a high probability, because the magnitude and distribution of graph entropy substantially reflect the structural and geometric properties of the graph. There have been successful applications exploiting graph entropy to estimate the weights of indexes in system evaluation. This indicates that graph entropy is capable of capturing the random property of local graph structure so as to be able to guide the selection of candidate resources for VNE.

1.2. Graph Entropy Measures

Graph entropy measures have been extensively applied in a variety of problem areas with multiple forms of definition, such as discrete mathematics, computer science, information theory, statistics, finance, computational biology, knowledge mining, structural chemistry, etc. [10]. A majority of applications characterize networks through graph entropy measures to quantify the structural information content of these networks for capturing their complexity. There graph entropy was applied for measuring the network complexity in a probabilistic approach, differentiating with deterministic methods such as Kolmogorov complexity measure using encoding and substructure approach counting the number of the specified substructures (see [11] for details).

Generally, the Shannon entropy corresponding to Boltzmann entropy in thermodynamics has the following form:

$$I_S = -\sum_k p_k \log(p_k), \tag{1}$$

which is a special form of the following Rényi entropy of order q in case of $q = 1$:

$$I_R = \frac{1}{1-q} \log\left(\sum_k p_k^q\right). \tag{2}$$

For measuring topological information, classical graph entropy measures, originally defined and explored by Rashevsky [12], Trucco [13] and Mowshowitz [14], use intrinsic structural features of a graph to determine a probability distribution over the graph. Usually, a set X of graph elements, called the graph invariant that means the cardinality of X being invariant under graph isomorphisms, along with an equivalence relation π which induces a partition of X into X_i, can define a probability distribution by letting $p_i = P(v \in X_i) = |X_i| / |X|$. Based on this, the Shannon entropy formula I_s can be applied to obtain a general definition of graph entropy as follows:

$$I(G) = -\sum_{i=1}^{k} \frac{|X_i|}{|X|} \log\left(\frac{|X_i|}{|X|}\right). \tag{3}$$

Particularly, Rashevsky [12] defines X as the set of vertices, namely $X = V(G)$, and X_i as the i-th vertex orbit of $V(G)$, where all orbits of $V(G)$ are generated by the vertex automorphism group of G. Trucco [13] introduces similar entropy measures by setting X as the edge set of G, namely $X = E(G)$, and X_i as the i-th edge orbit under edge automorphism group. Mowshowitz [14] define $X = V(G)$ and X_i as i-th chromatic decomposition of the vertices.

Körner [15] introduces the first definition of graph entropy called Körner entropy, using an extrinsic probability distribution (not necessarily induced by graph invariant). Graph entropy measures based on the partition of graph elements are not computable for large networks. Recently, Dehmer [16,17] proposed the concept of parametric graph entropy, in which information functions of capturing structural features of a graph are designed to derive probability distribution on graph vertices, and graph entropy is measured by Shannon formula. In particular, the following information functions have been proposed [11]:

$$f_1(v) = \alpha^{\sum_{k=1}^{\rho} c_k |s_k(v,G)|}, \ \alpha > 0, \ c_k > 0, \ 1 \le k \le \rho(G), \tag{4}$$

where $\rho(G)$ denotes the diameter of G;

$$f_2(v) = \sum_{k=1}^{\rho(G)} c_k |S_k(v,G)|, \ \alpha > 0, \ c_k > 0, \ 1 \le k \le \rho(G); \tag{5}$$

$$f_3(v) = |\lambda_v(G)|, \tag{6}$$

where λ_v denotes the eigenvalue indexed by vertex v of the adjacency matrix $A(G)$.

The graph entropies corresponding to these information functions can be uniformly defined by

$$I_f(G) = -\sum_{i=1}^{|V|} \frac{f(v_i)}{\sum_{j=1}^{|V|} f(v_j)} \log\left(\frac{f(v_i)}{\sum_{j=1}^{|V|} f(v_j)}\right). \tag{7}$$

The research of applying information entropy to measure the structure information of networks was initiated in 1979, when Bonchev et al. [18] provided a complete index survey aiming to measure chemical molecules and atoms. Since then, information entropy theory has been applied to society network research to find the potentially interesting substructure of objective social networks [19,20].

Distinctly, we would like to apply graph entropy theory to communication networks whose nodes indicate communication end devices, and links represent communication lines (virtual or real), instead of nodes indicating persons or organizations and links representing relations between nodes. For our purpose, we are particularly interested in applying graph entropy for efficient virtual network embedding. Thus it is reasonable to consider using parametric graph entropy for quantifying a network, given that an information function is flexible enough to describe the properties of the node resources and link bandwidth.

2. Definition and Model of VNE

In network virtualization, the user demand is presented as a virtual network (VN). In VN, the nodes and links indicate the resource and communication demands, respectively. The virtual network embedding (VNE) algorithm, designed by a virtual network provider (VNP), embeds VN into SN by way of resource allocation. The VNE scheme imposes a substantial impact to the performance of the NV system. Thus, an efficient VNE solution pays a critical role in NV technology. Theoretically, VNE can be modeled as a generalized map from VN to SN, by which the graph H abstracting VN is embedded into graph G representing SN. The embedding must satisfy some constrains over the requested and provided resources, and it should optimize some parameters of interest to the user and virtual network provider (VNP), such as maximal provider revenue and accepted ratio, and minimal embedding cost [1]. To gain an intuition for grasping these notions, consider a two-level architectural model for virtual network embedding depicted in Figure 1, where a series of virtual network requests (VNR), presented as virtual networks, have been embedded into two substrate networks operated by two infrastructures InP1 and InP2.

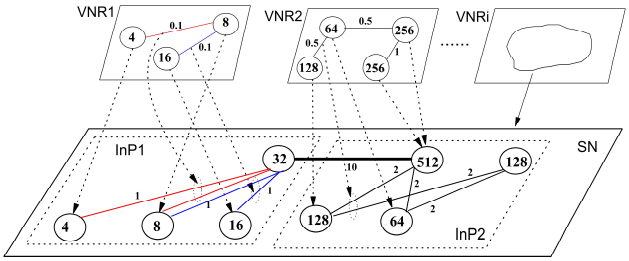


Figure 1. A two-level architectural model for virtual network embedding, with the correspondence between virtual edge and physical edge for InP2 omitted. The numbers in circles indicate the switching capacity of the routing and switching devices, and the ones near the links represent the transmission bandwidth, in Gbps. Virtual-to-physical edge correspondence is marked by distinct colors. VNR: virtual network request; SN: substrate network.

Finding the optimal solution to a general graph embedding with constrains is an NP-hard problem. Much research has dealt with designing heuristic algorithms to solve it [5–7], an area which has received much attention in recent years with the spread of network virtualization.

Let graph $G = (V_G, E_G)$ represent SN, where V_G denotes the set of physical nodes and E_G the set of physical links. Likewise, let $H = (V_H, E_H)$ represent a virtual network request (VNR) from user, where V_H denotes the set of virtual nodes and E_H the set of virtual links. Let $c(x)$ and $d(x)$ be two functions representing the available resource and demand of network entity x, respectively. Then the problem of embedding H into G can be modeled as finding functions f which are subject to $\forall x \in G$, $d(x) \leq c(f(x))$. If the objective is to minimize the cost of the embedding operation, the current known methods to find f can be characterized as solving the following optimization problem:

$$\arg\min_f \{cost(f) \mid cost(f)\} = \sum_{e \in E_H} \sum_{l \in f(e)} cost(d(l)) + \sum_{v \in V_H} \sum_{u \in f(v)} cost(d(u)), \quad (8)$$

where $cost(x)$ denotes the cost of the variable x.

3. VNE Algorithm Using Graph Entropy

3.1. Algorithmic Profile

For large substrate network G, the computational burden of searching resource will be relieved if candidates in substrate network can be confined in a particularly selected resource space, in which the distribution of resources exhibit a maximal similarity with the virtual network H. As an example, it can be observed in Figure 2 that embedding f_1 demonstrates an apparent dominance relative to embedding f_2 due to consideration of structural correspondence. The main idea of this work is to discover the structural similarity between substructures of substrate network G and the virtual network H through comparison of their graph entropies. Through graph entropy, the structure information of virtual and substrate networks can be quantified as respective entropies, which reflects the discrepancy of two compared structures, and narrows the space of objective solutions. More details regarding graph entropy measure can be found in [10,11,21].

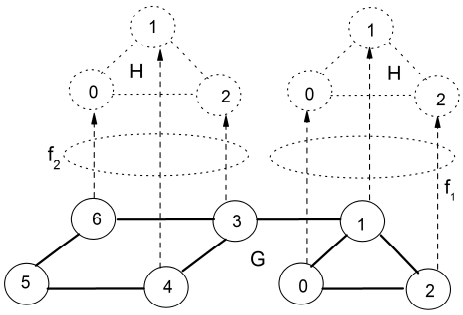


Figure 2. An example of embedding same one virtual network (VN) into two subnetworks.

The VNE algorithm based on graph entropy (GE-VNE) includes three procedures. In the first procedure, the algorithm searches a set of SN areas as candidates for VN embedding. These areas share common links or nodes. Then the algorithm detects one of the candidates with most resource as the optimal embedding area. In the second procedure, the algorithm searches all objects electable to fulfill the demands of VN in found candidates. These objects are structures simpler than those in the first procedure. In the final procedure, the graph entropies are calculated for all objects found in the second procedure, then the optimal candidate is found, and resource assignment will be implemented.

3.2. Selection of Candidate Areas

In the first stage of algorithm GE-VNE, for searching all SN substructures eligible to fulfill the virtual network request, procedures (described in Procedures 1 and 2) calculate the importance of all nodes. A natural way for completing this is to consider the quantity of node resource and the bandwidth of links incident to this node. Usually, SN nodes, holding more available resource and associating more available bandwidth, appear to be more important than those with less. Also, VN nodes are relatively important if they have high demands for resources. Consequently, the resource quantity of a node can indicate its importance (weights). Similarly, links transmitting heavy traffic mean large bandwidth demand in VN or resource in SN so that link bandwidth can characterize its weight. Because a virtual node can merely be embedded into a substrate node, the computation of graph entropy only involves in link attributes, regardless of node weights.

Then algorithm GE-VNE selects areas centered at nodes with higher importance in order as candidate Areas. In order to apply graph entropy for embedding virtual network, we define the importance of a node in network G as the sum of resource magnitude $c(v)$ plus the product of the number of links and the minimal bandwidth incident to v, namely,

$$w(v) = c(v) + |E(v)| \times \min\{c(l)|l \in E(v)\}, \tag{9}$$

where $E(v)$ denotes the set of links incident to v, and $c(v)$ denotes the available resource at entity v.

It seems to be reasonable that the best option area is the one with resource distribution closest to the VN. When multiple candidate areas are available for a customized virtual network request. For realizing this, it needs to estimate the resource distribution of all candidates in SN for objective VN. Let S_i denote the i-th candidate of a VN, its quantity $a(S_i)$ of available resource can be defined as

$$a(S_i) = \sum_{l_i \in E(S_i)} a(l_i) + \sum_{u \in V(S_i)} a(u). \tag{10}$$

By the above formula, the quantity of available resource in S_i has been expressed as the sum of all available resource in nodes and links within candidate S_i. Our algorithm would select S_i with greatest value $a(S_i)$ as the candidate area. In order to confine the coverage of embedding H into G, let $\rho(G)$ and $r(G)$ denote the diameter and the radius of graph G, respectively. We define the radius $r(G)$ of searching SN resource as follows:

$$r(G) = w_0 d_0 + w_1 |V(H)| + w_2 |E(H)|, \tag{11}$$

where d_0 denotes the average of all SN node pairwise distances, w_0, w_1, w_2 assign initial distance, node and link weights, respectively.

3.3. Computation of VN and SN Entropies

Subsequently, the algorithm proceeds to choose the best suitable one for VN embedding among all candidates by measuring their graph entropies, as described in Procedure 3. For measuring the parametric graph entropies $I(H)$ and $I(G)$ proposed by Dehmer [16,17] (see Section 1.2 for details), it is required to define and calculate $f(v)$ for all $v \in V(G)$ firstly, where $f(v)$ is a function quantifying the local structural information at each node v. It is particularly essential that defining $f(v)$ to accurately quantize the structural information of graph G. There are a few of methods to define $f(v)$ [10,11], depending on concrete application scenarios. The most common way is to define $f(v)$ in terms of structural features around v, such as the node degree $\delta(v)$ and the number of paths across v. Apparently, a well-defined $f(v)$ affects substantially the computation of graph entropy. As a challenge still remained, quantifying finely local information burdens computation overhead, and coarsely quantified one may weaken the ability of characterizing the objective graph.

To define $f(v)$, denote substrate network with graph G defined as $G = (V, E, P)$, in which V and E indicate the traditional sets of nodes and edges, respectively. Additionally, P is a probability function defined over graph G. For $f(v)$ proposed by Dehmer in [16,17], it is necessary to know $S_k(v, G)$ in advance, where $S_k(v, G)$ represents a vertex subset of $V(G)$ containing nodes in distance k from v, is called the k-sphere of v regarding G, namely,

$$S_k(v, G) = \{u \in V \mid d(v, u) \leq r_k, 1 \leq k\}. \tag{12}$$

Defining $S_k(v, G)$ using large increment k hardly perceives the structure discrepancy. Reversely, a small increment k maybe leads to a high redundant computation. Generally, VN has relatively simpler structural attributes than SN in node and link distribution. Thus, $S_k(v, G)$ should be defined separately in VN and SN to avoid the problem described above. Additionally, the hidden node (marked as black solid circle) should be taken into account for graph entropy computation. As an example, Figure 3a shows that a VN of three-node circle will be embedded into a SN of four-node circle, and VN is divided as $S_1(v, H)$ and $S_2(v, H)$. In Figure 3b, SN is partitioned as $S_1(v, H) = \{v_5, v_6\}$ and $S_2 = (v_3, v_4)$. It is worth noting that candidate substructure for VN embedding contains a hidden node v_5 which should be considered in process of computing $f(v)$ to reflect the differences between candidates even if this hidden node is transparent to users.

Suppose that all values of bandwidth resources lie in interval $[a, b]$ (units: Mb/s). We firstly divide $[a, b]$ into i parts with same length. Then the radius $r_k(G)$ of sphere $S_k(v, G)$ can be set up as $r_k(G) = a + k(b - a)/i$. The value of $r_k(G)$ can be adjusted by taking different values of the parameter i for VN and SN. The radius of $S_k(v, H)$ can be assigned by a same approach, instead of parameters a and b representing demands rather than resources. For a weighted graph G, the distribution of node weights are tightly connected to the values of $|S_k(v, G)|$. If a node u is incident to links with high bandwidth in $S_k(v, G)$, it should contribute more to the whole network communication function, namely, the probability distribution P over G has a high value $P(u)$ at node u. Generally, the weight coefficients c_k of $S_k(v, G)$ are arranged as an increasing arithmetic series to express the relation between link bandwidth incident to node u and probability distribution P.

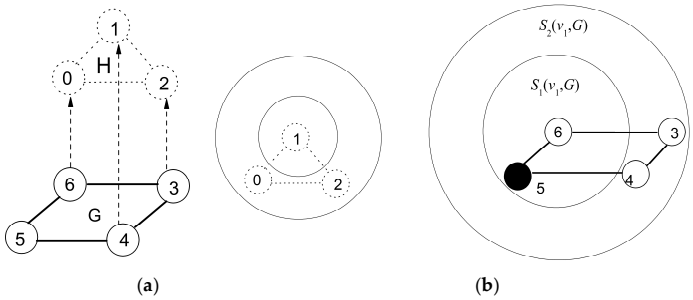


Figure 3. Entropy expression on structure information of VN and SN. (**a**) Structures of VN, SN, (**b**) Definitions of $S_k(v, G)$ and $S_k(v, H)$.

Then, the local information function $f(v)$ on vertex set V can be defined as

$$f(v) = \alpha^{\sum_{k=1}^{\rho} c_k |s_k(v,G)|}, \tag{13}$$

where α nd c_k are arbitrary and real positive coefficients, and $|S_k(v, G)|$ indicates the number of nodes located in interior of sphere centered at node v with radius k. Generally, one takes constants c_k $(1 \leq k \leq \rho)$ as an arithmetic sequence of positive integers. Letting $\rho_s(v) = \max_{1 \leq k \leq \rho} |S_k(v, G)|$, $\rho_c = \max_{1 \leq k \leq \rho} c_k$ and $r_c = \min_{1 \leq k \leq \rho} c_k$ to guarantee the graph entropy $I_f(G)$ derived from Formula (5) being bounded, the parameter α should satisfy inequality (see [17]):

$$1 < \alpha \leq |V|^{\frac{1}{\rho[\rho_s(v)\rho_c - r_c]}}. \tag{14}$$

To understand the process of graph entropy, Figure 4 instantiates a five-node network centered at node v_1, where all bandwidth resource lies in interval [10, 20]. Setting $i = 2$ it yields that $r_1(G) = 10$ and $r_2(G) = 20$ thus $S_1(v, G)$ and $S_2(v, G)$ are located at areas marked by large and small profiles of sphere, respectively. Further Setting up $c_k = k$, it yields $S_1(v_1, G) = \{v_2, v_3\}$ and $S_2(v_1, G) = \{v_4\}$ ($|S_1(v_1, G)| = 2$ and $|S_1(v_1, G)| = 1$). It follows from observation that $\rho(G) = 3$, $\rho_s(v_1) = 2$, $\rho_c = 2$, $r_c = 1$ and $\rho[\rho_s(v_1)\rho_c - r_c] = 9$. Thus the parameter α is bounded at $(1, 5^{1/9}]$ by Formula (14). Taking $\alpha = 5^{1/10} \in (1, 5^{1/9}]$, it yields $f(v) = 5^{1/10(1 \times 2 + 2 \times 1)} = 5^{0.4}$.

Once $f(v)$ is known, the function value $p_i = P(v_i)$ on vertex v_i of probability distribution P can be computed by formula:

$$p_i = \frac{f(v_i)}{\sum_{v_i \in V(G)} f(v_i)}, \tag{15}$$

Then the entropy $I(G)$ of graph G derives from the following expression in terms of probability distribution of G:

$$I(G) = -\sum_{v_i \in V(G)} p_i \log(p_i), \tag{16}$$

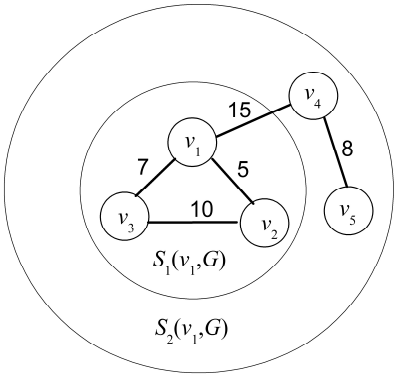


Figure 4. Computation of $S_k(v, G)$ in a five-node example network.

3.4. Presentation of Algorithmic Pseudocodes

Based on definitions and formulas presented above, we commence to describe the details of our algorithm. The relevant procedures may be further described as algorithmic pseudocodes, with corresponding step-by-step comments, listed as follows.

Algorithm 1. GE-VNE (Procedure 1): Embedding Areas Search

Input: SN, VN
Output: $\{S_i\}$
1. for each node $u_i (1 \leq i \leq n)$ in SN
2. calculate $w(u_i)$;
3. end for
4. sort $\{u_i \mid 1 \leq i \leq n\}$ as $\{\mu_i \mid 1 \leq i \leq n\}$ by $w(u_i)$ in descending order;
5. select p top important nodes $\{\mu_i \mid 1 \leq i \leq p\}$;
6. for each $\mu_i \in \{\mu_i \mid 1 \leq i \leq p\}$
7. $S_i \leftarrow S_k(\mu_i, G, r_k(G))$;
8. end for
9. $S_c \leftarrow S_1$;
10. for each S_i in $\{S_i \mid 1 \leq i \leq p\}$
11. calculate $a(S_i)$;
12. if $a(S_i) > a(S_c)$
13. $S_c \leftarrow S_i$;
14. end if
15. end for
16. output S_c as the candidate area for VN embedding.

In Procedure 1 (Algorithm 1):

Lines 1–3: calculate the importance of each node in SN by Formula (9);

Lines 4–5: select p top nodes $\{\mu_i \mid 1 \leq i \leq p\}$ according their importance;

Lines 6–8: for each of nodes $\{\mu_i \mid 1 \leq i \leq p\}$, calculate a network area centered in μ_i with radius r, as candidate areas for VN embedding, where r can be computed by Formula (11). Consequently, the embedding candidates for VN should have p areas: S_1–S_p.

Line 9–15: calculate the quantity $a(S_i)$ for each network area S_i, and select the largest one S_c.

Line12: output the selected embedding area S_c;

Algorithm 2. GE-VNE (Procedure 2): Embedding Candidates Searching

1. sort all virtual nodes by their importance as $V(H) = \{v_1, v_2, \ldots, v_n\}$;
2. sort all substrate nodes by their importance as $S = \{u_1, u_2, \ldots, u_n\}$;
3. initialize the candidate of VN as S_{11} = null
4. for each node v_i in $V(H)$
5. search top s nodes of SN which fulfill the demand of node v_i:
   $T_i = \{u_i \mid 1 \leq i \leq s, u_i \in V(G), c(u_i) \geq d(v_i)\}$
6. for each node u_j in T_i
   $S_{ij} = S_{(i-1)\mathrm{j}} \cup u_j$
7. end for
8. end for;
9. output S_{ij}

In Procedure 2 (Algorithm 2), as the central part of algorithm GE-VNE, search all SN substructures eligible to fulfill the virtual network request. Here introduce this procedure firstly, then an example is provided to facilitate understanding the implementation of GE-VNE.

In Figure 5, the VN node with largest importance is v_1, and all candidates for v_1 are $\{u_1, u_2, u_3\}$. If setting $s = 2$, it is readily observed that $T_1 = \{u_1, u_2\}$, $T_2 = T_3 = \{u_1, u_2, u_3\}$, and $S_1 = \{u_1\}$, $S_2 = \{u_2\}$; $S_{11} = \{u_1, u_1\}$, $S_{12} = \{u_2, u_1\}$, $S_{21} = \{u_1, u_2\}$, $S_{22} = \{u_2, u_2\}$, $S_{13} = \{u_1, u_3\}$, $S_{23} = \{u_2, u_3\}$; $S_{111} = \{u_1, u_1, u_1\}$, $S_{211} = \{u_2, u_1, u_1\}$, $S_{121} = \{u_1, u_2, u_1\}$, $S_{221} = \{u_2, u_2, u_1\}$, $S_{131} = \{u_1, u_3, u_1\}$, $S_{231} = \{u_2, u_3, u_1\}$; notably, the elements in sets here are assumed to be in order for corresponding to the ordered set $\{v_1, v_2, \ldots, v_n\}$ of virtual nodes, which makes $\{u_3, u_1\}$ and $\{u_1, u_3\}$ be different sets. Once procedure proceeds to its end, it will yield all expected candidates for objective VN.

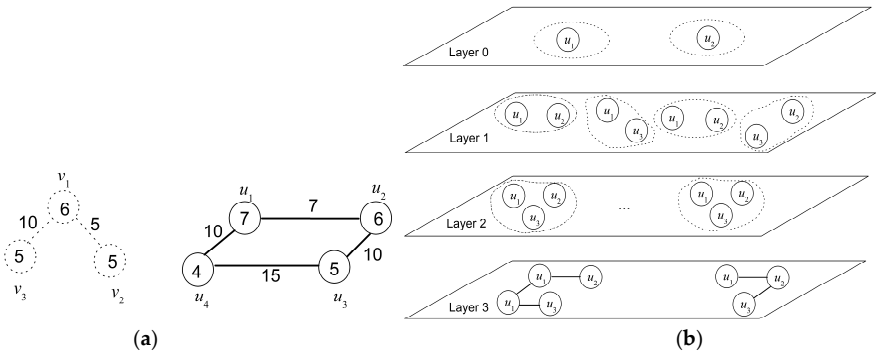


Figure 5. An example of finding a candidate of VN in SN, (**a**) VN (**left**) and SN (**right**), (**b**) searching candidates in SN.

Algorithm 3. GE-VNE (Procedure 3): VNE based on graph entropy

Input: $\{S_{c1}, S_{c1}, \ldots, S_{cr}\}, H$
Output: f: $H \rightarrow G$
1. calculate entropy $I(H)$ of H
2. for each S_{ci} in $\{S_{c1}, S_{c2}, \ldots, S_{cr}\}$
3. calculate entropy $I(S_{ci})$;
4. $d(I(S_{ci})) \leftarrow |I(S_{ci}) - I(H)|$
5. end for
6. sort $\{S_{c1}, S_{c2}, \ldots, S_{cr}\}$ as $\{T_{c1}, T_{c2}, \ldots, T_{cr}\}$ by $d(I(S_{ci}))$ as ascending order
7. for each T_{ci} in $\{T_{c1}, T_{c2}, \ldots, T_{cr}\}$
8. if (T_{ci} fulfills VN)
9. break;
10. end if;
11. end for

In Procedure 3 (Algorithm 3), algorithm GE-VNE chooses the best suitable one for VN among all candidates by measuring their graph entropies, as described in details listed as follows.

Line 1: calculate the entropy $I(H)$ of graph H;

Line 2–5: calculate the entropies $I(S_{ci})$ of graph series S_{ci}, and the entropy distances $|I(S_{ci}) - I(H)|$ between graphs S_{ci} and H;

Line 6: sort S_{ci} by entropy distance in ascending order;

Line 7–9: travel S_{ci} in order until finding one which fulfills the demand of H.

3.5. Limitations of This Work

The major limitation of algorithm GE-VNE is that entropy measures capture the structural attributes of different graphs efficiently, for example, the degree distribution. However, values from entropy measures are evidently inadequate in reflecting the topological discrepancy between graphs, particularly in cases of characterizing topological similarities on small graphs. The reason behind this can be understood readily from the definition of $f(v)$ in Formula (13), where two arbitrary nodes u and v yield identical functional value $f(\cdot)$ as long as $S_k(u, G) = S_k(v, G)$. This coincidence will happen in a high probability when the scales of objective networks are small. For example, Figure 6 exposes that two different graphs, (a) the Binary Tree BT_4 and (b) 2×2 Mesh $M(2, 2)$, might have the same entropy value though they are distinguished topologically, based on an observation that it holds $S_j(v_i, BT_4) = S_j(u, M(2, 2))$ $(1 \leq j \leq k, 1 \leq i \leq 4))$ for all nodes by setting same parameters α, c_k, $r_1(G) = 10$, $r_2(G) = 20$ and $k = 2$ on both graphs. Given that two adjacent virtual nodes may be mapped to two separate substrate nodes connected by a path, the optimization for the VNE problem sometimes advocates of charactering graphs through structural attributes rather than contrasting their topological similarity.

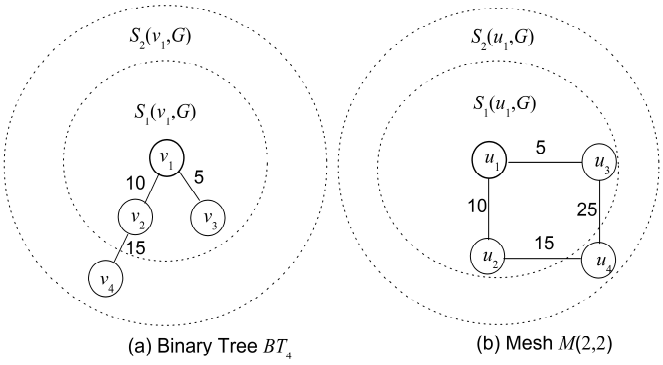


Figure 6. An example of exhibiting probability of equal entropies existing in two different topologies: (**a**) Binary Tree BT_4 and (**b**) 2×2 Mesh $M(2, 2)$.

4. Experiments

4.1. Experimental Configuration

In this section, we report on the results in a number of simulations conducted to experimentally validate performance and quality of algorithm GE-VNE. The experimental platform is facilitated with software IDE Eclipse (Neon, Eclipse Foundation, Ottawa, ON, Canada) under the 32-bit Windows 7 operating system (Microsoft Corporation, Redmond, WA, USA), and hardware CPU Intel(R) Core(TM) i7 5600-U @2.6 GHz with 8.0 GB RAM. All simulations generate the results with Alevin 2.1, developed by Beck et al. [22], that has successfully functioned as a simulation framework for examining virtual network embedding algorithms.

We encode the algorithm with programming language Java to generate subclass extending the class GenericMappingAlgorithm that has been realized as an algorithmic framework of generic VNE algorithms, and implemented all simulations under various scenarios. The experimental process

consists of network generation, algorithm configuration and execution, and algorithm evaluation, with various experimental configurations.

The experimental steps and corresponding configurations are further detailed in Tables 1 and 2. Also, a comparison with a couple of representative VNE algorithms, that have been cited as the focus of considerable VNE research, has been conducted in terms of runtime, VNR acceptance ratio, cost revenue ratio (cost/revenue), and node utilization ratio, such factors that have been recognized as effective factors of assessing VNE algorithms. Eventually, the results of comparison have been figured to perceive the performance and quality of algorithms in comparison.

Table 1. Parameters values used in GE-VNE.

Parameters	Values	Description
$r_k(H)$	0	Radius of $S_k(v, G)$ for VN
$r_k(G)$	20	Radius of $S_k(v, G)$ for SN
$r_j - r_{j-1}$	10	The increment of radius for $S_k(v, G)$
α	3	Constant in Formula (13)
c_i $(1 \leq i \leq 7)$	1–7	Weights in Formula (13)
p	6	Number of candidate areas
s	1/3	Number of candidate nodes
w_i $(1 \leq i \leq 3)$	1	Initial distance factor in Formula (9)

Table 2. Algorithmic parameters.

Parameters	Description	Values
dist	Distance for candidates	20
wCPU	CPU weight	1
wBw	Bandwidth weight	1
nodeoverload	Node overload concerned	false
type	Type of routing	0
k	Number of *k*-shortest path	5

- *Scenario Generation*

The theoretical analysis in the previous section focuses on using graph attributes for charactering graphs rather than topological association. This reminds us to conduct simulations paying less attention to the variety of network topologies. Thus two network topologies, the Binary Tree with 100 nodes (BT_{100}), and the 7×7 Mesh $M(2, 2)$, are selected as the SN models. Both networks have been recognized as practical topologies modeling datacenters in clouding computing environment and the Internet, also emerged as representative architectures for high performance computing, and appears to be cumulatively essential in era of undergoing multicore computer [23]. The step of network generation goes through establishing a network topology, adding resources to SN, as well as adding demands to VN. Additionally, VN topologies are randomly generated as 20 networks with sizes from 1 to 4. All SN node and bandwidth resources are randomly generated in interval [50, 100], and VN node and bandwidth demands are randomly generated in intervals [1, 20] and [1, 50], respectively (Units: Mb/s).

- *Algorithm Configuration*

Algorithms chosen for experimental evaluation involve five representative VNE algorithms, which have been proposed in [1,4,6] described in Table 3. These algorithms along with GE-VNE are executed under identical scenarios and parameter configurations. Evaluations are run 40–50 times in order to observe the performance of all algorithms. In the stage of mapping nodes, the weights of CPU nodes 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 stage of mapping links,

the parameter k of mapping a VN link to a length-k shortest path is set to $k = 2$. The other parameters pertaining to our algorithm are listed in Table 2.

Table 3. Representative VNE approaches chosen for evaluation, the citation times updated on 8 July 2017.

Algorithm	Reference	Brief Description
DViNE-SP DViNE-PS	Chowdhury et al. [2]	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 [24] 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 [24] citations: 998
RW-MM-SP RW-MM-PS	Cheng et al. [7]	VNE ranking nodes with topology properties for node mapping and k-shortest paths for link mapping. Google Scholar [24] citations: 346

4.2. Experimental Results

In order to evaluate performance of the algorithm GE-VNE, six metrics with respect to performance are considered in our experiment: average links stress (ALS); VNR acceptance ratio (AR); cost/revenue ratio (CRR); cost; link utilization (LU); and link cost per VNR (LCPV). The acceptance ratio reflects the fraction of VNRs successfully embedded as virtual networks. The revenue sums the revenue of the VNRs that were successfully mapped and the revenue of those that were not mapped. The cost measures the quantity of substrate resources allocated for VNR. The link utilization reflects the proportion of bandwidth being utilized to meet the currently accepted VNRs. The concrete implications of these VNE evaluation metrics have been interpreted in Table 4. Three conclusions can be observed from simulations.

Table 4. Descriptions of VNE evaluation metrics.

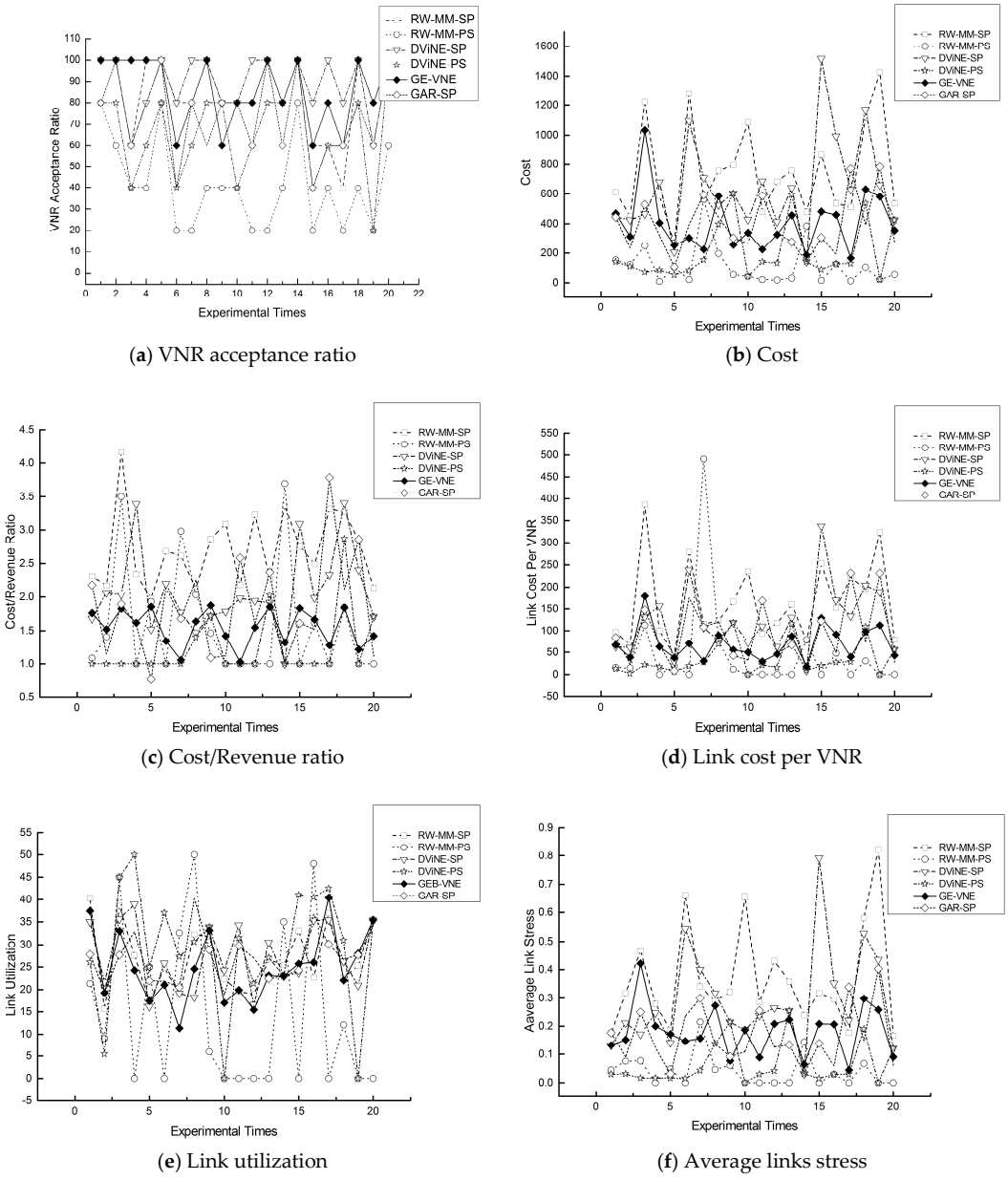
Metrics	Interpretations
ALS	The average of the proportion of occupied bandwidth on each link
AR	The ratio between the number of accepted VNRs and the total number of VNRs
CRR	The ratio of embedding cost and revenue
cost	The sum of the substrate resources allocated to the VNR
LU	The proportion of occupied bandwidth
LCPV	The link cost for embedding each VN

In respect to the VNR acceptance ratio, Figure 7a displays that GE-VNE behaves comparatively better to other algorithms; also observed in Table 5, the VNR acceptance ratio caused by GE-VNE increases 41.25% and 59% to DViNE-SP and DViNE-PS, respectively. In Figure 8a, GE-VNE leads to a moderate VNR acceptance ratio as executing it on substrate network BT_{100}.

Regarding three cost-related metrics: cost, cost/revenue, and average link cost, Figures 7 and 8b–d indicate that algorithm GE-VNE yields relative low values to other algorithms as implementing them on $M(7, 7)$ and BT_{100} for 20 times, also seen in Table 5. An extra should be observed in mentioned figures that GE-VNE holds a higher value than RW-MM-PS and DViNE-PS in average link cost, but latter two algorithms lead to a pretty low acceptance ratio 15% and 40.75%, respectively. Only algorithm DViNE-PS approaches GE-VNE in VNR acceptance ratio, but the former spends link cost 154.61, considerably higher than 84.83 by GE-VNE. Therefore, GE-VNE reduces the link cost in the situation of increasing VNR acceptance ratio. A promotion in whole embedding cost emerging in Figure 7b, is attributed to higher cost for node embedding for higher VNR acceptance ratio.

Figures 7 and 8e,f illustrate improvements in link utilization and the average stress for 20 times VN embedding on experimental scenarios of embedding a random VN into 7×7 Mesh $M(7, 7)$ in Figure 7 and BT_{100} in Figure 8. By definition, high link cost or VNR acceptance causes link stress increasing, which emerges in Figure 7f because of appearing a high VNR acceptance in GE-VNE, also seen in Table 5.

As depicted in both Figures 7 and 8, GE-VNE earns a better trade-off in VNR acceptance and link stress than DViNE-SP, and overall, reduces the average stress of each embedded VN to the whole substrate network on both $M(7, 7)$ and BT_{100}.



(**a**) VNR acceptance ratio (**b**) Cost

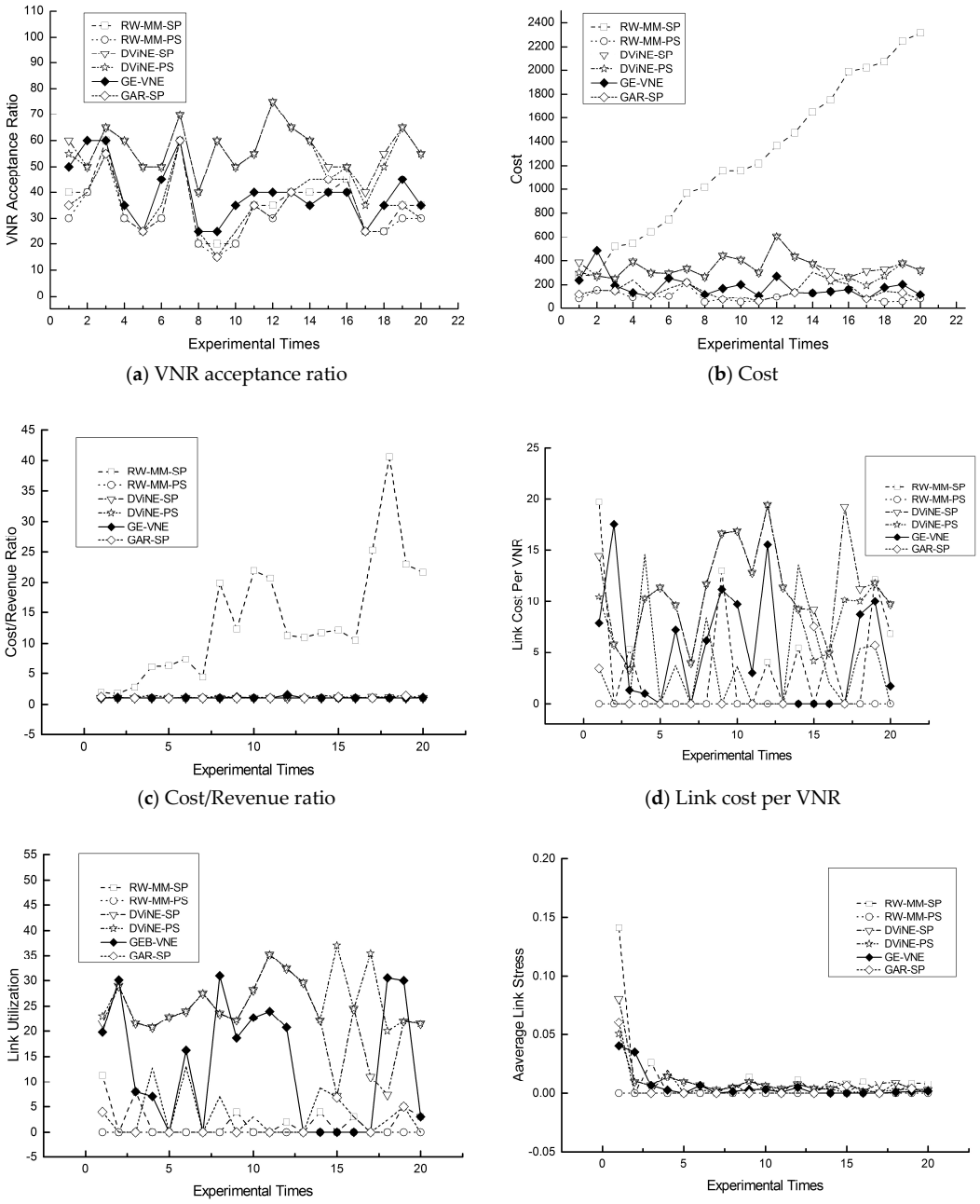
(**c**) Cost/Revenue ratio (**d**) Link cost per VNR

(**e**) Link utilization (**f**) Average links stress

Figure 7. The comparisons of GE-VNE with other representative VNE approaches in terms of six VNE metrics, (**a**) AR; (**b**) Cost; (**c**) CRR; (**d**) LCPV; (**e**) LU; and (**f**) ALS, from executing algorithms 20 times on 7×7 Mesh $M(7, 7)$.

Table 5. Comparison of average results from executing algorithms 20 times on 7×7 Mesh.

Algorithms	AR	Cost	CRR	LCPV	LU	ALS
RW-MM-SP	84%	10.50	2.70	163.80	10.5	0.37
RW-MM-PS	41%	750.7	1.58	44.80	26.8	0.04
DViNE-SP	89%	116.0	2.07	122.75	14.2	0.29
DViNE-PS	66%	642.7	1.24	34.84	27.5	0.063
GE-VNE	86%	203.8	1.55	68.70	28.7	0.18
GAR-SP	74%	402.8	1.82	91.71	24.9	0.16



(**a**) VNR acceptance ratio (**b**) Cost

(**c**) Cost/Revenue ratio (**d**) Link cost per VNR

(**e**) Link utilization (**f**) Average links stress

Figure 8. The comparisons of GE-VNE with other representative VNE approaches in terms of six VNE metrics, (**a**) AR; (**b**) Cost; (**c**) CRR; (**d**) LCPV; (**e**) LU; and (**f**) ALS, from executing algorithms 20 times on BT_{100}.

5. Conclusions

The particularly essential part for virtual network embedding is to choose an optimal subnetwork as candidate of the virtual network to be embedded. Previous VNE algorithms mainly concern the optimization of node and link embedding relative to some anticipated parameters, neglecting the impact of substrate network structure on VNE.

We propose a VNE algorithm based on graph entropy called GE-VNE to extract and quantify the structure information for both VN and SN. GE-VNE divides SN into substructures, and considers multiple available candidate structures for embedding objective VN, then searches all resources on SN

to find the richest one as embedding area of VN rather than using traditional graph partition. Then the algorithm extracts available substructures within some distance constraint as candidates. Finally, the optimal candidate is found by comparison of graph entropies of all candidates.

Experimental results on the Alevin Platform have shown that our algorithm exhibits some merits regarding a group of principal VNE evaluation metrics. It was also observed that there is no dominance of our algorithm on some of these metrics. Future research has been scheduled to explore improvements to this problem.

Author Contributions: J.Z. performed initially the theoretical derivation, experiments, and wrote the draft of this paper in Chinese; C.Z. proposed the original idea, checked the initial derivation, revised radically the paper, and rewrote the paper in English; H.W. made a major contribution to the revision of this paper as for the review comments received. M.L. analyzed the experimental data. R.D. conceived and designed the experiments.

Funding: This research was funded by [National Science Foundation of China] grant number [61562089].

Acknowledgments: This research is supported by National Science Foundation of China, Grant No. 61562089.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

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

References

1. Fischer, A.; Botero, J.F.; Beck, M.T.; de Meer, H.; Hesselbach, X. Virtual Network Embedding: A Survey. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 1888–1906. [CrossRef]
2. Chowdhury, N.; Rahman, M.R.; Boutaba, R. Virtual network embedding with coordinated node and link mapping. In Proceedings of the IEEE INFOCOM, Rio de Janeiro, Brazil, 19–25 April 2009; pp. 19–25.
3. Jarray, A.; Karmouch, A. Decomposition approaches for virtual network embedding with one-shot node and link mapping. *IEEE/ACM Trans. Netw.* **2015**, *23*, 1012–1025. [CrossRef]
4. Chen, X.; Li, C.; Jiang, Y. Optimization model and algorithm for energy efficient virtual node embedding. *IEEE Commun. Lett.* **2015**, *19*, 327–1330. [CrossRef]
5. 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*, 17–29. [CrossRef]
6. 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 (VISA '09), Barcelona, Spain, 17 August 2009; pp. 81–88.
7. Cheng, X.; Su, S.; Zhang, Z.; Wang, H.; Yang, F. Virtual network embedding through topology-aware node ranking. *ACM SIGCOMM Comput. Commun. Rev.* **2011**, *41*, 38–47. [CrossRef]
8. Beck, M.T.; Fischer, A.; Demeer, H. Distributed and scalable embedding of virtual networks. *J. Netw. Comput. Appl.* **2015**, *56*, 124–136. [CrossRef]
9. Zhang, S.; Qian, Z.; Wu, J.; Lu, S.; Epstein, L. Virtual Network Embedding with Opportunistic Resource Sharing. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 816–827. [CrossRef]
10. Dehmer, M.; Mowshowitz, A. A history of graph entropy measures. *Inf. Sci.* **2011**, *181*, 57–78. [CrossRef]
11. Mowshowitz, A.; Dehmer, M. Entropy and the complexity of graphs revisited. *Entropy* **2012**, *14*, 559–570. [CrossRef]
12. Rashevsky, N. Life information theory and topology. *Bull. Math. Biophys.* **1955**, *17*, 229–235. [CrossRef]
13. Trucco, E. A note on the information content of graphs. *Bull. Math. Biol.* **1956**, *18*, 129–135. [CrossRef]

14. Mowshowitz, A. Entropy and the complexity of graphs IV: Entropy measures and graphical structure. *Bull. Math. Biophys.* **1968**, *30*, 533–546. [CrossRef]
15. Körner, J. Coding of an information source having ambiguous alphabet and the entropy of graphs. In Proceedings of the 6th Prague Conference on Information Theory, Prague, Czech Republic, 19–25 September 1971; pp. 411–425.
16. Dehmer, M. Information processing in complex networks: Graph entropy and information functionals. *Appl. Math. Comput.* **2008**, *201*, 82–94. [CrossRef]
17. Dehmer, M. A novel method for measuring the structural information content of networks. *Cybern. Syst.* **2008**, *39*, 825–842. [CrossRef]
18. Bonchev, D.; Balaban, D.; Mekenyan, A.T. Generalization of the graph center concept, and derived topological centric indexes. *J. Chem. Inf. Comput. Sci.* **1980**, *20*, 106–113. [CrossRef]
19. Emmert-Streib, F. The chronic fatigue syndrome: A comparative pathway analysis. *J. Comput. Biol. A J. Comput. Mol. Cell Biol.* **2007**, *14*, 961–972. [CrossRef] [PubMed]
20. Emmert-Streib, F.; Dehmer, M. Gobal information processing in gene networks: Fault tolerance. In Proceedings of the 2nd Bio-Inspired Models of Network, Information and Computing Systems (Bionetics 2007), Budapest, Hungary, 10–12 December 2007; pp. 3009–3028.
21. Dehmer, M.; Chen, Z.; Li, X.; Shi, Y. *Mathematical Foundations and Applications of Graph Entropy*; Wiley-Blackwell: Oxford, UK, 2016; pp. 101–112.
22. Beck, M.T.; Linnhoff-Popien, C.; Fischer, A. A simulation framework for Virtual Network Embedding algorithms. In Proceedings of the 2014 16th International Telecommunications Network Strategy and Planning Symposium, Funchal, Portugal, 17–19 September 2014; pp. 1–6.
23. Leiserson, C.E. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Trans. Comput.* **2012**, *C-34*, 892–901. [CrossRef]
24. On-Line Data. Available online: https://scholar.google.com (accessed on 1 April 2017).