Computer Communications 80 (2016) 1–15

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

A modified ACO algorithm for virtual network embedding based on graph decomposition

Fangjin Zhu, Hua Wang*

School of Computer Science and Technology, Shandong University, PR China

ARTICLE INFO

Article history:
Received 17 October 2014
Revised 22 May 2015
Accepted 11 July 2015
Available online 17 July 2015

Keywords:
Network virtualization
Virtual network embedding
Ant colony optimization algorithm
Virtual network topology invariance
Topology decomposition

ABSTRACT

Network virtualization is a promising scheme to solve Internet ossification. A challenging problem in this scheme is virtual network embedding (VNE), which involves efficiently embedding multiple heterogeneous virtual networks into one or more physical networks. The VNE problem is known to be NP-hard and thus requires an approximate algorithm as a solution. This study models the VNE problem based on virtual network topology invariance and analyzes the shortcomings of a general embedding algorithm under different network topologies. A modified ant colony optimization algorithm is proposed based on network topology decomposition. A pre-computation algorithm is first proposed based on ant random walking to accelerate the recognition of the ring characteristics of a network topology. Pre-computation results are used to guide the decomposition of virtual networks and the embedding process of ring structures. The topology of a virtual network is decomposed into a combination of ring structures and tree structures, which have different characteristics. Different embedding algorithms are then designed for these structures. Point-disjoint paths are searched for any two virtual links to ensure the reliability of the network topology in the embedding process. The proposed algorithm shows an enhanced optimization performance, which is better than those of the ViNE-LB and GN-SP algorithms.

1. Introduction

An increasing number of applications with various qualities of service (QoS) requirements are being developed over the Internet. However, supporting different data delivery mechanisms suitable for these QoS requirements is difficult for traditional TCP/IP models. This problem is referred to as Internet ossification [1]. As a promising scheme to address this ossification problem, network virtualization is proposed to build a diversified Internet that supports a variety of network services and architectures through a shared substrate by decoupling network functionalities and infrastructures [2].

In this scheme, traditional Internet service providers (ISPs) are separated into infrastructure providers (InPs) and service providers (SPs). InPs maintain network resources, and SPs lease network resources to construct their own virtual networks (VNs). InPs need to embed VNs into their physical networks by mapping one physical node for a virtual node and searching one loop-free path for a virtual link. Thus, many VNs of different SPs would coexist in a physical network. Different mapping methods result in different resource consumption. Hence, InPs face the important problem of efficiently mapping these VNs into a physical network to gain the highest revenue. This problem is referred to as the virtual network embedding (VNE) problem.

The VNE problem involves both node mapping and link mapping. A well-known NP-hard multiple separator problem can be reduced to the VNE problem [3,4]. Therefore, the VNE problem is known to be NP-hard. Single node or link mapping can also be NP-hard, and solving them requires heuristic and approximate algorithms. Several methods have been proposed to address the VNE problem [5]. These techniques are summarized into the following categories: two-stage mapping methods [4,6–9], coordinated node and link mapping methods [10–12], and mapping methods based on graph theory [6,11–14]. However, common problems in research need to be discussed and addressed.

(1) ***General embedding algorithms are used for different topologies of VN requests***. Some of the aforementioned methods employ a general mapping technique for any VN requests and ignore the differences among VN topologies [4,6–12]. These differences always indicate different key relations that affect the embedding process (details are discussed in Section 3). A general algorithm cannot reflect these differences efficiently.
(2) ***Existing embedding algorithms based on graph decomposition are too simple***. Other methods attempt to reduce the

* Corresponding author. Tel.: +86 53188061653.
E-mail addresses: zhufj@sdu.edu.cn (F. Zhu), wanghua@sdu.edu.cn (H. Wang).

http://dx.doi.org/10.1016/j.comcom.2015.07.014

complexity of mapping algorithms and propose some techniques based on graph decomposition and combination. In some works [6], network topology is divided into combinations of star structures, but the hardness of closed characteristics remains. In other works [13], network topology is divided into core networks and edge networks, and different existing mapping algorithms are used. However, the algorithms for mapping core and edge networks still fail to consider different types of topologies.

(3) ***Existing embedding algorithms may decrease the reliability requirement of SPs***. The topology of a VN always implicitly contains some requirements of the owners of VNs (SPs), such as the requirement of redundancy degree. However, most mapping methods ignore this requirement, resulting in different virtual links sharing one or more physical links or nodes [4,6–11]. This phenomenon can decrease the reliability of a VN, and some mutual backup virtual links in the design may fail simultaneously.

We propose an embedding algorithm that guarantees that the mapping networks generated in a substrate network (SN) satisfy the reliability requirements of SPs. The proposed algorithm attempts to improve the acceptance ratio while maximizing the revenue of an InP. The following are our main ideas and contributions to the VNE problem:

- ***Improved optimization performance for InPs***. To improve the optimization performance, a pre-computation algorithm is designed to obtain the ring characteristics of a graph (a VN request topology or SN topology). A VN topology is randomly divided into a combination of ring structures and tree structures by using the ring characteristics of the VN request. Different construction algorithms are then designed for these structures. An algorithm for ring structures is designed based on the ring characteristics of the SN. An algorithm for tree structures is designed based on tree-indexed random walking.
- ***Realization of reliability requirements of SPs***. To maintain the reliability of a VN, the concept of candidate node is proposed to design point-disjoint paths for the virtual links in the mapping process. In this way, a node (or a link) is prevented from being shared by different virtual links of the VN.
- *Convenience of resource management and protocol deployment for InPs*. Resource management and protocol deployment in VNs are eventually realized in SNs. In our algorithm, the unique disjoint SN path for each virtual link facilitates explicit resource management (such as resource reservation) and is convenient for deploying special data transmission protocols (such as multi-protocol label switching or MPLS) in SNs. In existing works, the sharing of nodes or links among SN paths may cause routing loops, which result in extra mechanisms (such as mechanisms for avoiding packet disorder). This will make resource management and protocol deployment increasingly difficult for InPs.

A modified ant colony optimization (ACO) algorithm is introduced to improve optimization performance. The ACO algorithm is a population-based algorithm, inspired by the foraging behavior of ants and proposed by Dorigo [15]. Ants communicate with one another through pheromone; based on this, the ACO algorithm uses pheromone to accelerate knowledge in problem solving; this algorithm has been successfully applied in many combination optimization problems [16]. In our algorithm, each ant randomly constructs a complete solution according to pheromone and heuristic information. The solutions are then evaluated with a fitness function that guides them toward optimization through multiple iteration processes.

Simulation results show that the idea of random selection in tree structure mapping (random root selection and random selected subtree growing) and ring structure mapping (random adjacent node selection) results in a flexible and sufficient graph traversal. Moreover, the ACO algorithm improves optimization performance. Thus, reliability is guaranteed, and high acceptance rate and revenue are obtained.

This paper is organized as follows. Section 2 summarizes related works on the VNE problem. Section 3 describes the background of the VNE problem and defines the optimization problem to be addressed. Section 4 proposes a modified ACO algorithm to solve the optimization problem. Section 5 discusses the simulation results and analysis. Section 6 concludes the study.

2. Related works

The VNE problem is an NP-hard problem with different optimization objectives; it involves node mapping and link mapping [5]. In the first part of this section, we mainly summarize existing works with the same objective as that of our study. In the second part, other works with objectives that differ from that of our study to a certain degree are introduced. Lastly, the relations between these works and our work are analyzed.

2.1. Related works with the same optimization objective

The objectives of the works in [4,6–14,17–22] are the same as that of the present study, i.e., to minimize the cost of a VN request, or maximize the revenue of an InP. In these studies, the cost of a VN request is generally computed as the weighted sum of the CPU costs of all mapped nodes and the bandwidth costs of all mapped paths; thus, the general solutions attempt to determine the mapped SN nodes that are close to one another and to identify the shortest SN path between any two mapped nodes if a virtual link exists between them in a VN request [6–9], or use multiple paths that simultaneously provide bandwidth for a virtual link if the SN supports splitting data transmit mechanisms [4,10]. To realize these objectives, some "sufficient capacity" nodes with more nodes and edges, or more CPU and bandwidth resources, in their neighboring area, are determined. "Sufficient capacity" nodes imply that their neighboring areas have more opportunities to embed other nodes and links of a VN request.

Based on the aforementioned greedy idea, some forms of heuristic information are defined with several network topology attributes [6–9], and a single path (the shortest or the k-shortest path) is used for a virtual link. In [6], Zhu et al. proposed a basic VN assignment scheme, which defined neighbor resource availability as heuristic information to select nodes greedily, and used the shortest path for each virtual link. In [7], Cheng et al. referenced the Markov random walk model and proposed 'NodeRank' as a measurement for node resource based on this model. The NodeRank value of a node was used as heuristic information, and two kinds of algorithms were designed: RW-MaxMatch (a two stage algorithm), and RW-BFS (a coordinated node and link mapping algorithm). To improve the performance of their two stage algorithm, they used a particle swarm optimization (PSO) algorithm to search for the optimal solution among all feasible solutions. In [8], Li et al. proposed a two stage scheme, which used multiple factors of a network topology that included hop count as a top-k dominating model to greedy node mapping and used k-shortest path to link mapping. In [9], Cui et al. proposed an algorithm based on the maximum convergence degree, to ensure that the topology of a virtual network gathered together when it was mapped onto an SN, and the k-shortest path was used for link mapping. In these works, heuristic information could improve the performance, but load balance was not considered. The load balance of an SN is beneficial to accepting successive VN requests and gaining additional revenues. However, load balance with a single path for a virtual link is NP-hard [4].

To reduce the computation complexity caused by single paths, a multi-commodity flow (MCF) concept was introduced in some works [4,10], which allowed a virtual link to be mapped to multiple SN paths. Moreover, the corresponding linear (or mixed integer) programming models based on the MCF concept were proposed. In [4], the amount of node and link resources of an SN node was defined as heuristic information, and the nodes were mapped greedily. Moreover, a linear programming (LP) model was proposed and solved with existing tools. Unlike in [4], a mixed integer programming (MIP) model was introduced in [10] by augmenting an SN. A relaxed mechanism was used to handle the hardness caused by integer variables, and each SN node gained a probability value to map a virtual node. A node was selected by using a rounding technique deterministically or randomly, and then, the LP model was solved again for link mapping. Enhanced performance can be realized in these algorithms, by introducing load balancing. However, the support of splitting data transmission in an SN may translate into additional costs in resource management and protocol deployment.

Except for the use of MCF, some algorithms based on graph theory (mainly graph decomposition) have been proposed to reduce the hardness of the VNE problem [6,11–14]. In [6], Zhu et al. proposed an improved VN assignment algorithm, which divided a large VN request into a series of star topologies. Thus, acceptance ratio could be improved, and link and node stress could be balanced. In [12], Lischka et al. used a relaxed subgraph isomorphism detection algorithm to find a feasible solution in one stage and used the hop count constraint to minimize the cost of a VN request. Enhanced optimization performance is achieved by using candidates set as Cartesian product of all nodes in the node selection process. In [11], Fajjari et al. proposed a scheme for dividing a large VN request into several smaller sub-requests (solution components) based on access nodes with fixed geographic locations. This work also used some topology attributes such as “hanging links” to define heuristic information. In [13], Qing et al. divided a VN into core and edge networks. In [14], Huang et al. presented some ideas for mapping different topologies based on the “importance” of the virtual nodes in their structures. However, concrete algorithms are not proposed in these works. Although graph decomposition can reduce the complexity of the mapping process by mapping smaller partitions of a VN request separately, the combination of these partitions is not trivial. Moreover, the mapping sequences of different partitions and within these partitions may severely influence final optimization performance.

To improve the optimization performance that is restricted by the mapping sequence in previous works, some meta-heuristic algorithms are proposed. With the help of randomization and evolving multiple iteration processes, enhanced optimization performance is achieved. In [7,11,17,18], some algorithms based on PSO, ACO, and artificial fish swarm were proposed to search for an optimal solution among candidate solutions. In [19], several meta-heuristic based algorithms were discussed and compared based on certain metrics and enhanced optimization results were obtained from these algorithms, particularly in ACO-based algorithms.

Other studies differ from the aforementioned works but have the same optimization objective. In [20], a path algebra-based strategy was used to optimize link mapping phase. All possible feasible paths were listed and ordered after all virtual nodes were mapped. Paths were then selected from these paths to construct a feasible solution to the VNE problem. In [21], an online embedding framework which was adapted from the online primal-dual model was proposed. Moreover, the VNE problem was formulated as a linear program, and an online algorithm was designed to solve the dynamic linear programs. A competitive result was obtained even when the sequence of VN requests was not known in advance. In [22], a framework was proposed to address the dynamics of each VN request itself, namely, the increase or decrease of virtual nodes and the requirement changes of VN nodes or links. Moreover, some heuristic algorithms were proposed to address these sub-problems.

2.2. Related works with optimization objectives that differ to a certain degree

Some works with optimization objectives that differ from those of our schemes also exist. In [23], a distributed algorithm was proposed; its objective was to determine an effective trade-off between minimizing a VN request cost and minimizing communication cost. In [24], Beck et al. presented a distributed, parallel, and generic VNE framework based on hierarchical network partitioning. They achieved lower message overhead and kept embedding costs comparable with those of centralized approaches. In [25], the authors proposed a VNE algorithm in a software-defined network environment. In addition to minimizing the cost of a VN request, the authors also attempted to minimize communication delays between the central controller and the distributed controllers in each SN node.

References [26] and [27] focused on the VNE problem in multiple SN networks. Maximizing revenue across multiple SNs was considered, and the emphasis was placed on the cooperation policies and mechanisms among multiple SNs. Refs. [28] and [29] focused on the reliable VNE problem. Their optimization objectives emphasized supporting a reliable VN by taking over some redundant lines, while minimizing the provision cost of each VN request.

Refs. [30] and [31] focused on the green VNE problem (or energy-aware VNE problem). In these studies, minimizing active nodes and links to minimize energy consumption while embedding all VN requests was discussed.

Refs. [32] and [33] discussed the reconfiguration of the VNE problem to overcome the fragmentation of SN resources that was caused by the incoming and outgoing VN requests, which eventually resulted in poor performance. The authors discussed how the nodes and links that need to be reconfigured were selected, and enhanced performance was achieved with the reconfiguration scheme.

In [34], research issues and challenges in wireless network virtualization were summarized and discussed. Future research directions were also proposed.

2.3. Relations between our work and existing works

Our work mainly focuses on minimizing VN cost while improving the acceptance ratio in an SN. Inspired by existing works, we use graph decomposition to solve the VNE problem. We also use the randomization and evolving iteration processes of meta-heuristic algorithms to improve optimization performance. Moreover, some greedy ideas, such as selecting “sufficient capacity” nodes greedily in each iteration process, were employed. The greedy ideas adopted in our algorithm indicate that the heuristic information defined in existing works can be used complementarily with our work. However, the difference of our work with existing works is that we distinguish the meaning of adjacency (or close) node between a ring structure and a tree structure, as well as design different mapping algorithms for these structures. To guarantee the implications of a VN request, we introduce the concept of VN topology invariance, which also results in convenient resource management and protocol deployment. We also use candidate nodes to realize this objective. The simulation results in synthetic and real network topologies verify that enhanced optimization performance is achieved in our algorithm even if a much-restricted constraint of VN topology invariance is introduced.

3. Background and notations

The VNE problem is described through an example. The SN and VN requests are modeled as undirected graphs, and a mathematical model based on some definitions and notations is presented.

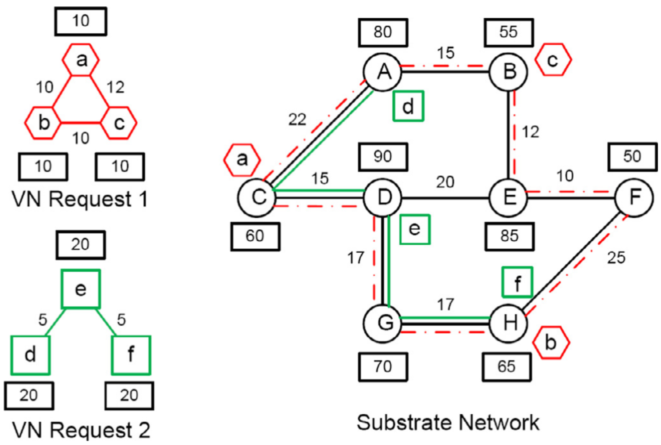

Fig. 1. Substrate network (right) and virtual network requests (left) [10].

3.1. Background and network notations

Both the SN and VN requests are modeled as weighted undirected graphs. An SN is modeled as $G^S = (V^S, E^S)$, where S denotes SN, V^S denotes the set of nodes, andE^S denotes the set of edges of the SN. A VN request is modeled as $G^R = (V^R, E^R)$, where R denotes the VN request. We use $C^{S,V}_{attr}(v)$ to denote the vertex attribution *attr* of a substrate node $v \in V^S$ and use $C^{S,E}_{attr}(e)$ to denote the edge attribution *attr* of a substrate edge $e \in E^S$, where S denotes the SN, and V and E denote vertex and edge, respectively. The value of *attr* denotes an attribution of a vertex or an edge; for example, $C^{S,V}_{cpu}(v)$ is used to denote the *cpu* resource of a substrate node v, and $C^{S,E}_{bw}(e)$ is used to denote the bandwidth resource of a substrate edge e. For a virtual node $v \in V^R$ and a virtual link $e \in E^R$, the donations are $C^{R,V}_{attr}(v)$ and $C^{R,E}_{attr}(e)$, respectively. In Fig. 1 [10], the *cpu* resource $C^{S,V}_{cpu}$ of a substrate node (or *cpu* requirement $C^{R,V}_{cpu}$ of a virtual node) is placed in a rectangular frame near the node, and the bandwidth resource $C^{S,E}_{bw}$ of a substrate edge (or bandwidth requirement $C^{R,E}_{bw}$ of a virtual link) is a number placed directly along the edge.

Fig. 1 [10] shows the mapping results of VN requests 1 and 2. For VN request 1, the virtual nodes a, b, c are mapped to the substrate nodes C, H, B, respectively. The virtual links (a, b), (a, c), and (b, c) are mapped to the substrate paths (C, D, G, H), (C, A, B), and (C, E, F, H). The *cpu* resources of these substrate nodes satisfy the *cpu* requirements of the corresponding virtual nodes. The bandwidth resources of these substrate paths satisfy the bandwidth requirements of the corresponding virtual links. Therefore, VN request 1 is accepted, this mapping result is used, and the resources of the substrate nodes and edges are reduced accordingly. For the same reason, VN request 2 is also accepted because the residual resources of the SN can satisfy the resource requirements of the virtual nodes and links through the given mapping.

A VN request may have several mappings, each of which can generate different levels of resource consumption, which influence the acceptance of other VN requests. The revenue and cost of a mapping are first given to describe the influence of different mappings on the InP.

3.2. Revenue and mapping cost and VN topology invariance

For a VN request R, the revenue of all mappings is the same. Revenue is defined as the weighted sum of both the CPU resource requirements of all virtual nodes and the bandwidth requirements of all virtual links.

$$Revenu(R) = \alpha^* \sum_{v \in V^R} C^{R,V}_{cpu}(v) + \beta^* \sum_{e \in E^R} C^{R,E}_{bw}(e) \tag{1}$$

where α, β reflect the relative importance of CPU resources and bandwidth, respectively, to the revenue.

The cost of a VN request depends on the different mappings onto the SN. Cost is defined as the weighted sum of both the CPU resource requirements of all virtual nodes and the reserved bandwidth of all virtual links. The reserved bandwidth of a virtual link is defined as the reserved bandwidth of a substrate path, which is the mapping of this virtual link. Thus, the cost of mapping a VN request is defined as follows:

$$Cost(R) = \alpha^* \sum_{v \in V^R} C^{R,V}_{cpu}(v) + \beta^* \sum_{e \in E^R} C^{R,E}_{bw}(e)^* \left\| P(e) \right\| \tag{2}$$

where $p(e)$ is the mapped substrate path of a virtual link e, and $\| p(e) \|$ is the hop length of the path $p(e)$. The path $p(e)$ is determined by a VNE algorithm, and $\| p(e) \|$ is the inflation degree of a virtual link. The maximum value of the inflation degrees of all virtual links is defined as the inflation degree of a VN and can be expressed as follows:

$$\delta = \max_{e \in E^R} \left(\left\| p(e) \right\| \right) \tag{3}$$

The value of δ always reflects the delay requirement of a VN request and is a relaxed isomorphism constraint of a mapping; $\delta = 1$ indicates the need for a strict (isomorphic) substrate mapping [36] of a VN request, and each virtual link is mapped to a substrate edge. In this case, the cost of a VN request is the same as its revenue. $\delta > 1$ indicates that a relaxed substrate mapping of a VN request can be used. A virtual link can be mapped to a substrate path whose hop length is larger than 1. In this case, the endpoints of the substrate path can satisfy the CPU resource requirements of the two corresponding virtual nodes. The minimum bandwidth of all the substrate edges of the mapped substrate path can then satisfy the bandwidth requirement of the virtual link. A strict mapping ($\delta = 1$) is sometimes difficult to find, and a relaxed mapping ($\delta > 1$) is always used in the VNE algorithm to improve the acceptance ratio. A long mapped substrate path is beneficial in obtaining the load balance of the whole SN and in avoiding the shared substrate edge of different virtual links.

In this work, the mapped substrate paths of any two virtual links must not share a (or some) substrate edge (edges). The reason for this constraint is that the VN topology is defined by an SP and indicates some requirements of the SP. For example, two or more paths existing between two virtual nodes reflects the reliability requirement of the SP network; when a path fails, the other paths can guarantee the connectedness of the VN. We call this constraint the *VN topology invariance*. To achieve this characteristic for our VNE algorithm, some special constraints are introduced to our mathematical model of the VNE problem.

(1) When searching a substrate node for a virtual node, the substrate node must satisfy two constraints: the residual CPU resource of this substrate node must not be lower than the CPU resource requirement of the virtual node, and the residual degree of this substrate node must not be lower than the degree of the virtual node.

The residual CPU resource of a substrate node v is defined as

$$R_{cpu}(v) = C^{S,V}_{cpu}(v) - \sum_{v' \mapsto v : v' \in V^R : VNRaccepted} C^{R,V}_{cpu}(v') \tag{4}$$

where the second part of the formula is the sum of the CPU requirements of all virtual nodes mapped to the node v of all accepted VN requests.

The residual degree $R_{degree}(v)$ of a substrate node v is defined as the total degree of reduction in the degrees used by the other links of the current VN request.

The first constraint is the same as that in the literature, that is, a virtual node must be located in one substrate node; however, the second constraint is generally ignored, except [12]. The degree constraint is a necessary condition for maintaining the characteristic of the VN topology invariance.

(2) To search for a loop-free substrate path for a virtual link, two constraints must be satisfied: the minimum residual bandwidth of this path must not be lower than the bandwidth requirement of the virtual link, and any two paths must be guaranteed to be point-disjoint paths.

The minimum residual bandwidth of a path is defined as the minimum residual bandwidth of all substrate edges that belong to this path.

$$\mathrm{R}_{bw}(p(e)) = \min\left(\mathrm{R}_{bw}(e_1),\ \mathrm{R}_{bw}(e_2), \ldots, \mathrm{R}_{bw}(e_n)\right) \tag{5}$$

where $p(e) = (e_1, e_2, \ldots, e_n), e_1, e_2, \ldots, e_n \in E^S$ is the loop-free substrate path for the virtual link $e \in E^R$, and the residual bandwidth of a substrate edge is defined as

$$R_{bw}(e) = C_{bw}^{S,E}(e) - \sum_{e' \mapsto e : e' \in V^R : VNRaccepted} C_{bw}^{R,E}(e') \tag{6}$$

To prevent any two mapped paths from sharing the same substrate edges or nodes, we use the following constraints:

$$point - disjoint(p_1, p_2) \tag{7}$$

Two loop-free paths are referred to as point-disjoint paths when these two paths do not have the same intermediate points. This constraint can guarantee that the two paths do not share an intermediate point or an intermediate link.

The first constraint of a path is the same as that in the literature, that is, a virtual link to a path that satisfies the bandwidth requirements must be located. The second constraint of a path is to avoid any two paths sharing the same substrate edges or nodes. The reason for the second constraint is also to guarantee that the VN topology invariance maintains the reliability of the VNE algorithm.

According to these introduced constraints and the traditional constraints of the VNE problem, the VNE problem is mathematically modeled in the following section.

3.3. Mathematical model of VNE problem based on VN topology invariance

A mapping of a VN request to an SN can be described as $f : G^R \to G^S$; each virtual node $v \in V^R$ is mapped to a substrate node $f(v) \in V^S$, and each virtual link $e = (u, v) \in E^R$ is mapped to a loop-free substrate path $f(e) = p(f(u), \ldots, f(v))$. To guarantee the success of this mapping, the constraints of the nodes and paths described in the previous section must be satisfied.

The load balance of the whole SN should be also considered to improve the acceptance ratio of our VNE algorithm. A balanced SN indicates that a high acceptance ratio can be obtained. The utility percentage of a substrate node is defined as

$$P_{uti}(v) = \frac{C_{cpu}^{S,V}(v) - \mathrm{R}_{cpu}(v)}{C_{cpu}^{S,V}(v)} \tag{8}$$

The utility percentage of a substrate edge is defined as

$$P_{uti}(e) = \frac{C_{bw}^{S,E}(e) - \mathrm{R}_{bw}(e)}{C_{bw}^{S,E}(e)} \tag{9}$$

The variance of the utility percentage of the nodes and edges of the SN is used as the load balance factor.

$$LB(G^S) = \sqrt{\frac{1}{m-1}\sum_{v \in V^S}\left(P_{uti}(v) - \frac{1}{m}\sum_{v \in V^S} P_{uti}(v)\right)^2 + \frac{1}{n-1}\sum_{e \in E^S}\left(P_{uti}(e) - \frac{1}{n}\sum_{e \in E^S} P_{uti}(e)\right)^2} \tag{10}$$

where $m = |V^S|$ is the number of substrate nodes, and $n = |E^S|$ is the number of substrate edges.

To assess how different inflation degrees influence acceptance ratio and cost or reflect the delay constraint of a VN request, an inflation degree constraint is given: for any path p, $\|p\| \le \delta$.

Based on the above analysis, the mathematical model of our VNE algorithm for guaranteeing VN topology invariance is given by

$$Min \quad \alpha Cost(R) + \beta LB(G^S) \tag{11}$$

$s.t.$

$$\forall v \in V^S, \quad \sum_{i \in V^R} S_{i,v} \le 1 \tag{12}$$

$$\forall i \in V^R, \sum_{v \in V^S} S_{i,v} = 1 \tag{13}$$

$$\forall v \in V^S, i, k \in V^R, \quad \sum_{j \in E^R} X_{j,v} \begin{cases} \le 1, & \text{if} \sum_{i \in V^R} S_{i,v} = 0 \\ = Deg(k), & \text{if} \sum_{i \in V^R} S_{i,v} = 1 \text{and} S_{k,v} = 1 \end{cases} \tag{14}$$

$$\forall j = (k, l) \in E^R, k, l \in V^R, v_1, v_2 \in V^S, \quad \sum_{v \in V^S} X_{j,v} \begin{cases} = 2, & \text{if} S_{k,v_1} = 1, S_{l,v_2} = 1, (v_1, v_2) \in E^S \\ >2, \le \rho+1, & \text{if} S_{k,v_1} = 1, S_{l,v_2} = 1, (v_1, v_2) \notin E^S \end{cases} \tag{15}$$

$$\forall v, u \in V^S, \forall j \in E^R, \quad \sum_{u \in N(v)} X_{j,u} = \begin{cases} 1, & \text{if} S_{i,v} \cdot X_{j,v} = 1 \\ 2, & \text{if}(1 - S_{i,v}) \cdot X_{j,v} = 1 \\ 0, & \text{if} X_{j,v} = 0 \end{cases} \tag{16}$$

$$\forall i \in V^R, \quad \forall v \in V^S, \quad C_{cpu}^{R,V}(i) \cdot S_{i,v} \le \mathrm{R}_{cpu}(v) \tag{17}$$

$$\forall j \in E^R,\ \forall u, v \in V^S,\ (u, v) \in E^S, X_{j,u} \cdot X_{j,v} \cdot C_{bw}^{R,E}(j) \le \mathrm{R}_{bw}(u, v) \tag{18}$$

$$S_{i,v} \in (0, 1) \tag{19}$$

$$X_{j,v} \in (0, 1) \tag{20}$$

where $\rho$is the inflation factor (the hop count of a mapped substrate path) which provides a constraint to the length of a mapped path for a virtual link, $Deg(v)$ is the degree of a node v, and $N(v)$ is the neighboring nodes set of a node v.

In the preceding mathematical model, we define two binary matrices S and X. Binary matrix S denotes the mapping relations between virtual nodes and substrate nodes. Moreover, if a virtual node $i \in V^R$ is mapped to a substrate node $v \in V^S$, then element $S_{i,v} = 1$, else $S_{i,v} = 0$. Matrix X denotes the mapping relations between virtual links and substrate nodes. Moreover, if a virtual link $j \in E^R$ uses a substrate node $v \in V^S$ (the substrate node is on the mapped path of the virtual link), then element $X_{j,v} = 1$, else $X_{j,v} = 0$. When the feasible values of these two matrices are obtained, the mapping relationship between a VN request and an SN is decided. Moreover, the optimization objective is used to evaluate the solution. However, the values must satisfy some constraints to reflect our ideas. In particular, VN topology invariance must be guaranteed in the mapping. We introduce the meaning of the optimization objective and each constraint in the following.

For each VN request R, Eq. (11) simultaneously minimizes both the cost of the mapped SN and the load balance factor of the entire SN, thereby reflecting our greedy idea of accepting successive VN requests, and α and β reflect our trade-off between these two optimization goals. Meanwhile, Eq. (12) and (13) guarantee that each virtual node is mapped to one substrate node, and neither node is mapped to the same substrate node. In addition, Eq. (14) indicates that when a substrate node is not a mapping of any virtual node, this substrate node is only used for one virtual link at most, which is a constraint to guarantee all mapped paths that must be disjointed. If not, the substrate node is used once by each virtual link and one endpoint is mapped to this substrate node. Hence, the total number is equal to the degree of the corresponding virtual node. Based on Eq. (15), when a virtual link is mapped to a substrate edge, two substrate nodes are used; otherwise, some intermediate nodes must be used for this virtual link; and path length is limited to ρ(the number of nodes that is used by a mapped path for a virtual link is not larger than $\rho+1$). Meanwhile, Eq. (16) exerts the constraint that for each substrate node used by a virtual link, if it is an endpoint of this virtual link, then one of its associated edges is used, and if not, two of its associated edges must be used. This constraint, which is associated with that of Eq. (15), guarantees that a loop free path is used for each virtual link. Along with the constraint in Eq. (14), these constraints are proposed to search disjoint loop free paths for all virtual links, which can guarantee VN topology invariance. Meanwhile, Eq. (17) exerts the constraint that the residual CPU resource of the mapped substrate node can satisfy the CPU requirement of a virtual node, whereas Eq. (18) guarantees that the residual bandwidth of the mapped substrate path can satisfy the bandwidth requirement of a virtual link.

The preceding mathematical model involves the integer variables of two binary matrices, and this model is intractable if the scale of VNE problem is large [35]. Thus, a modified ACO is used in our paper.

3.4. Fitness function of the proposed VNE algorithm

The ACO algorithm [15,16] is an iterated algorithm, and a fitness function to evaluate a solution is necessary to guide the following iteration process. Based on the aforementioned mathematical model and analysis, a suitable solution is that with minimum substrate cost and improved load balance. This solution should satisfy the constraint of the VN topology invariance. However, this constraint is violated at the stage of merging the different parts that correspond to the sub-graphs of the VN into a whole network. Improved solutions would evolve from these invalid solutions. A trade-off between invariance characteristics and probabilistic opportunity is needed to obtain enhanced solutions; thus, the number of shared nodes and links is also introduced into the fitness function as a penalty factor to select some promising solutions. The fitness function is defined as

$$f(s) = \frac{\alpha}{Cost(R)} + \frac{\beta}{LB(G^S)} + \frac{\Delta}{\varepsilon + num} \tag{21}$$

where num is the number of shared nodes and links, $\varepsilon > 0$ is a small real constant to provide the function with meaning when num is zero, and α, β, Δ adjust the relative importance among all the factors. A solution with larger fitness value is considered as a better solution.

4. ACO algorithm for the VNE problem based on topology decomposition

In this section, we analyze the difference between ring topology and tree topology in the mapping process. We also point out that general embedding algorithms based on "random walking" do not reflect such difference and that different mapping ideas and methods are thus needed. We then propose a mapping algorithm based on topology decomposition, in which the VN topology is first decomposed into a combination of a series of ring topologies and tree topologies. Different mapping methods are then proposed for these topologies to map substrate sub-networks. The mapped substrate sub-networks are then merged into one mapping solution.

Different processing sequences may result in different decomposition results, which may lead to different mapping solutions and thus influence mapping success. Random topology decomposition and random walking are adopted to increase the flexibility of our algorithm. These random ideas are consistent with the idea of the random solution construction of the ACO algorithm. Thus, a modified ACO algorithm is proposed to realize these random ideas easily. Adopting the evolutional iteration processes of the ACO algorithm in our modified algorithm can help us obtain satisfactory performance.

4.1. Differences between ring topology and tree topology in terms of mapping process

In the modified ACO algorithm, different mapping methods are adopted to map ring and tree topologies. In this section, we analyze the differences between these topologies using a general random walking construction algorithm. A virtual node is randomly selected as the start node. A substrate node that satisfies the resource requirement of the virtual node is also randomly selected as the mapping of this virtual node. Subsequent virtual nodes are selected one by one according to a certain defined sequence (wide first or deep first traverse). An "adjacent" substrate node for each virtual node is then searched until all virtual nodes are mapped or until the mapping is considered to have failed. In this algorithm, the adjacent node can satisfy the resource requirements of the virtual node, and the connecting substrate link can satisfy the bandwidth requirement of the corresponding virtual link.

The definition of "adjacent" varies in tree topology and ring topology because the former is a diffused topology, whereas the latter is a closed one. When searching the adjacent node for a tree topology, only the distance between the searching node and the last selected node is considered. When searching the adjacent node for a ring topology, other relations are considered in addition to the distance between the searching node and the last selected node. The reason for this consideration is that in a ring (closed) topology, the distance between the last node and the first node is also a part of the whole network cost. Given this difference, different mapping algorithms are needed for these two topologies when addressing the VNE problem.

Recognizing the ring characteristic of a substrate topology is important in efficiently mapping a ring topology request for cost minimization and given the constraint of VN topology invariance. A pre-computation algorithm for finding the ring characteristic of a topology is given in the next section.

4.2. Pre-computation algorithm for the ring characteristic of a topology

The probability of generating a ring among nodes, called *ring characteristic of a topology*, is first determined to find a ring topology from a general topology. In our algorithm, this probability is described by a data structure, which is provided by a pheromone table. This data structure is set up and maintained through the random walking of ants. This pre-computation can be used in a VN topology to guide the decomposing process and in the SN topology to guide the mapping process of each virtual ring topology.

Each node d maintains a pheromone table (τ_{ij}), where each row is a destination node that denotes a node in the network, each column denotes a neighbor node with a link to d, and τ_{ij} denotes the probability of generating a ring between the current node d and the destination node i through the neighbor node j. Some nodes cannot generate a ring given a certain neighbor node; therefore, $\tau_{ij} = 0$ is valid. In some cases, a node is not on any ring; therefore, all $\tau_{ij} = 0$ is also valid.

The pre-computation process is the initial process of our algorithm. In this process, the ant colony searches the ring characteristic of a network topology by cooperatively updating the pheromone table in each node. When the initial process is finished, the pheromone tables in all nodes can reflect the ring characteristic of the network topology.

Each node periodically sends an ant whose destination is the current node. In each node, the ant randomly selects a neighbor node as the next hop at a uniform distribution and records the passing nodes. When the ant reaches node k, it checks whether this node is in its passed path. If the node is not in its passed path, then it continues. If node k is in its passed path, a ring is detected, and all the nodes in this ring are obtained through the ant's record. For each node in this ring, the pheromone table is updated using the following rules: the destination nodes are the nodes in this ring, the neighbor node is the next node in the ring, and the corresponding pheromone value of these pairs (destination node and neighbor node) is increased by a certain value. After the pheromone tables are updated, the ant continues its walk beginning at node k using another neighbor that has not been used as the next node. The destination is still the node that sends the ant. The ant terminates its walk when it reaches the destination or when it has used up all neighbor nodes in one node. The first condition denotes that the ant has found a ring and that the pheromone tables in all the nodes in the ring are to be updated. In this case, the ant is deleted. The second condition implies that the ant cannot return to the start node, in which case the ant is also deleted.

After the initial process is finished, the ring characteristics of the network topology are distributed in the resulting pheromone tables in all nodes. We can then use this knowledge to guide the decomposing or mapping process.

4.3. VNE algorithm based on topology decomposition

In addressing the VNE problem, an inflation factor $\delta > 1$ allows a virtual link to be mapped to a loop-free substrate path whose hop length is larger than 1. All intermediate substrate nodes in this path are called candidate nodes, which are used in the final solution. To maintain VN topology invariance, any two virtual links are mapped in such a way that they do not share a common substrate node or link. To achieve this objective, we use a random construction algorithm and label all candidate nodes in this path as used; such nodes will no longer be used in the same VN request mapping process.

To guide the iteration process of an ant colony, each virtual request also has a pheromone table (ϕ_{ij}) associated with it. In this table, each row is a virtual node, and each column is a substrate node; ϕ_{ij} is the probability of virtual node i mapped to substrate node j.

Each node of the SN maintains a local statistic structure that records the usage of resources, including the initial CPU resource, the residual CPU resource of the current node, and the initial and residual bandwidths of each edge that connects with the current node. These values can be used as heuristic information in the mapping process.

4.3.1. Mapping algorithm for a ring structure

An SN first executes the pre-computation process to obtain its ring characteristics in support of the ring topology mapping process. The resulting pheromone table (τ_{ij}) of the pre-computation process in each node is used to guide the ring topology mapping process.

The solution construction process of an ant is described below.

For each virtual ring structure, a virtual node v_0 is randomly selected as the first node, all the other nodes in this ring are arranged clockwise or counter-clockwise $(v_1, v_2, \ldots, v_m)$, and the virtual nodes are mapped one by one by this order. For the first node v_0, a substrate node u_0 is selected randomly as its mapping in those substrate nodes that satisfy the resource requirements of the virtual node (Section 3). The next virtual node v_1 is obtained, and the unused neighbor nodes $n_1, n_2, \ldots n_N$ of node u_0 are checked. The corresponding pheromone $\tau_{u_0,n_j} > 0$ of node u_0 and the residual bandwidth of the link (u_0, n_j) satisfy the bandwidth requirement of the virtual link (v_0, v_1). One of these neighbor nodes is then selected randomly as the first candidate node c_1 for the virtual node v_1 and then labeled as used. The ant then proceeds to the candidate node c_1 and checks whether its residual resources (CPU resource and degree) can satisfy the resource requirements of the virtual node v_1. If such requirements are met, then this node is used as the final mapping; otherwise, the above process is performed again until a node u_1 is obtained as the mapping for this virtual node v_1. The candidate nodes $c_1, c_2, \ldots, c_k$ in path (u_0, u_1) to construct the mapping for the virtual link (v_0, v_1) will not be used any more for the current VN request. In this process, if a node does not have this kind of neighbor nodes, then the construction process is stopped. The process is repeated until all virtual nodes are mapped.

We assume that the virtual node v_{i-1} has been mapped to the substrate node u_{i-1}. For brevity, we denote u_{i-1} as c_0 for the virtual node v_i (c_0 is not really a candidate node of the current virtual node but the start node of its mapping process). The probability of selecting an unused neighbor node n_j of the kth candidate node c_k as the $(k+1)th$candidate node c_{k+1} for the virtual node v_i is defined as

$$p(c_{k+1} = n_j) = \frac{{\phi_{v_i,n_j}}^{\alpha} {\eta_{n_j}}^{\beta} {\tau_{u_0,n_j}}^{\gamma}}{\sum\limits_{n_j \in N_{c_k}} {\phi_{v_i,n_j}}^{\alpha} {\eta_{n_j}}^{\beta} {\tau_{u_0,n_j}}^{\gamma}} \tag{22}$$

where N_{c_k} is the set of unused neighbor nodes of the substrate node u_{i-1} (c_0) or the candidate node c_k in the path (u_{i-1}, u_i), and η_{n_j} is the heuristic information, which is defined as

$$\eta_{n_j} = \frac{\mathrm{R}_{bw}(c_k, n_j)}{\mathrm{C}_{\mathrm{bw}}^{\mathrm{S,E}}(c_k, n_j)} \tag{23}$$

According to Eq. (22), the probability of an SN node to be selected as a candidate node of a virtual node is decided based on three components: pheromone value, heuristic information, and ring characteristic value. Pheromone value indicates the probability of a virtual node mapping to an SN node, which is updated in previous iteration processes. A higher pheromone value implies that in the previous iteration processes, numerous ants use the SN node as the mapping of the virtual node, and the SN node has high probability of being selected as the mapping of the virtual node in the following iteration process. Thus, the algorithm is provided with opportunities to obtain an effective solution by exploiting the neighboring area of the SN node. Heuristic information indicates that an SN node connected to an SN edge with large bandwidth is selected at a high probability. This observation is based on the concept of load balance. The ring characteristic value reflects the probability of the SN node to form a ring with the initially mapped SN node, which is computed in the pre-computation process. A higher ring characteristic value indicates increased chances to form a ring. If the value is zero, the SN node cannot be selected as a candidate node. The values of α, β, γ reflect the relative importance of these three components respectively.

The mapping algorithm for a ring structure is presented as follows.

Input: A virtual ring structure $G^R = (V^R, E^R)$, a substrate network $G^S = (V^S, E^S)$, the pre-computation result (τ_{ij}) in each substrate node
Output: $f: v \in V^R \mapsto u \in V^S, f: (v_{i-1}, v_i) \in E^R \mapsto (u_{i-1}, c_1, \ldots, c_k, u_i)$, or fail (reject the virtual request)
1. Randomly select a virtual node v_0, and order all the nodes as $v_0, v_1, v_2, \ldots, v_m$ in a clockwise or counter-clockwise direction;
2. Randomly select a substrate node u_0 as the mapping of v_0;
3. for each virtual node v_i,
    Denote u_{i-1} as c_0;
    $k = 0$;
    Repeat
        Check all the unused neighboring nodes of c_k, find all the nodes $n_1, n_2, \ldots, n_N$ whose corresponding value $\tau_{u_0,n_j} > 0$, and $R_{bw}(c_k, n_j) \geq C^{R,E}_{bw}(v_{i-1}, v_i)$;
        Randomly select a neighboring node n_j as its $(k+1)th$ candidate node c_{k+1} using the probability defined in Eq. (22), and label it as used;
        if the resource of c_{k+1} can satisfy the requirement of the virtual node v_i,
        then $f: v_i \mapsto c_{k+1} = u_i, f: (v_{i-1}, v_i) \mapsto (u_{i-1}, c_1, \ldots, c_k, u_i)$, continue;
        else
            $k++$;
            if $k > \delta$, break;
            Endif
        Endif
    until u_i is found or $k > \delta$
Endfor
4. Output the mapping or failed results.

4.3.2. Mapping algorithm for a tree structure

This algorithm is also called a modified random tree-indexed mapping algorithm because we use the virtual sub-tree (a node with its child nodes) structure as index. In each step of our algorithm, a mapped virtual node whose child nodes have not been mapped is selected randomly; all its child nodes are mapped in one step. This step is repeated until none mapped nodes have unmapped child nodes. This algorithm differs from the deep-first or width-first traversal algorithm, which results in a fixed node sequence and proceeds the nodes in a linear order. The random sequence of sub-trees makes our algorithm flexible for the match between the VN request topology and the SN topology. In this way, the acceptance probability of a tree structure is improved.

The process of mapping all the child nodes of the selected node in each step is described as follows.

Assume that the selected virtual node is v and that it has been mapped to the substrate node u. The child nodes of node v are $v_1, v_2, \ldots, v_m$, and the links between v and v_i are denoted as l_i. As used in Section 3.3.1, 0th candidate node $c_{i,0}$ for each virtual node $v_i (i = 1, 2, \ldots, m)$ is defined as $c_{i,0} = u$. The unused neighboring nodes of node $u(c_{i,0})$ are $u_1, u_2, \ldots, u_n$, and the links between $u(c_{i,0})$ and u_j are denoted as e_j. According to our mapping model, $n \geq m$. We then map these virtual nodes to the SN.

First, $v_1, v_2, \ldots, v_m$ is sequenced in a decreasing order based on the bandwidth requirement of $l_i (i = 1, 2, \ldots, m)$. These nodes are processed in this order. For each child node v_i, a node u_j is randomly selected as its candidate mapping node $c_{i,1}$, which can satisfy the condition $C^{R,E}_{bw}(l_i) \leq R_{bw}(e_j)$. The candidate nodes are labeled as used. If a virtual node cannot find a candidate node, the algorithm is deemed to have failed.

For thekthcandidate mapping node $c_{i,k} (k = 1, 2, \ldots, \delta - 1)$ of each virtual node $v_i (i = 1, 2, \ldots, m)$, if $C^{R,V}_{cpu}(v_i) \leq R_{cpu}(c_{i,k})$ and $Dev(v_i) \leq R_{degree}(c_{i,k})$ are satisfied, then kth candidate node $c_{i,k}$ is the finial mapping node $f(v_i) = c_{i,k}$. If one of the two conditions is not satisfied, the $(k+1)th$ candidate mapping node $c_{i,k+1}$ is selected from the unused neighboring nodes of $c_{i,k}$. This process is repeated until a final mapping node is found for each virtual node or until the algorithm is considered to have failed.

The aforementioned algorithm aims to map the child nodes of a selected virtual node in one step. A substrate mapping tree is obtained when all the virtual nodes are mapped. Given that this algorithm is a random algorithm and fails sometimes, it must be run a number of times to obtain an entire solution. We use an ant colony with each ant executing this algorithm once and with some of the ants finding a mapping solution.

The probability of selecting an unused neighboring node u_j of the kth$(k = 0, 1, 2, \ldots, \delta - 1)$ candidate node $c_{i,k}$ as the $(k+1)th$ candidate node $c_{i,k+1}$ of the virtual node v_i is defined by the combination of the pheromone value and the heuristic information. The pheromone value ϕ_{ij} records the knowledge of the last iterations of an ACO algorithm to find suitable solutions. A node with a high pheromone value is selected with high probability. Meanwhile, heuristic information reflecting the status of a node should also be used as a selection component. The heuristic information is defined based on the local statistic structure that records the initial bandwidth and the residual bandwidth of a substrate edge e_j.

$$\eta_{u_j} = \frac{R_{bw}(e_j)}{C^{S,E}_{bw}(e_j)} \tag{24}$$

The candidate node selection probability is defined as follows:

$$p(c_{i,k+1} = u_j) = \frac{{\phi_{v_i,u_j}}^{\alpha} {\eta_{u_j}}^{\beta}}{\sum\limits_{u_j \in N_{c_{i,k}}} {\phi_{v_i,u_j}}^{\alpha} {\eta_{u_j}}^{\beta}} \tag{25}$$

where $N_{c_{i,k}}$ is the set of the unused neighboring nodes of the $k$thcandidate node $c_{i,k}$, and α, β reflect the relative importance of pheromone and heuristic information in the selection process.

The mapping algorithm for a tree structure is presented as follows.

Input: A virtual tree structure $G^R = (V^R, E^R)$, a substrate network $G^S = (V^S, E^S)$
Output: $f: v \in V^R \mapsto u \in V^S, f: (v_{i-1}, v_i) \in E^R \mapsto (u_{i-1}, c_1, \ldots, c_k, u_i)$, or fail (reject the virtual request)
1. Randomly select a virtual node as the root of the virtual tree structure;
2. Randomly select a substrate node as its mapping;
3. repeat
    Randomly select a mapped virtual node v whose child nodes have not been mapped;
    Sequence all the child nodes $v_1, v_2, \ldots, v_m$ in a decreasing order based on the bandwidth requirement;
    for each virtual node v_i,
        $k = 0$;
        $c_{i,k} = u = f(v)$;
        Randomly select an unused neighboring node of the kth candidate node $c_{i,k}$ as the $(k+1)th$ candidate node $c_{i,k+1}$ for each virtual node v_i, which satisfies $C^{R,E}_{bw}(l_i) \leq R_{bw}(e_j)$;
    endfor;
    for each node v_i,
        $k = 1$;
        while $(!C^{R,V}_{cpu}(v_i) \leq R_{cpu}(c_{i,k})$or $!Dev(v_i) \leq R_{degree}(c_{i,k}))$
            Randomly select an unused neighboring node u_j of $c_{i,k}$ as the $(k+1)$th candidate node $c_{i,k+1}$ based on the probability defined in Eq. (25), which also satisfies $C^{R,E}_{bw}(l_i) \leq R_{bw}(e_j)$;
            $k++$;
    Endwhile
        $f(v_i) = c_{i,k}$;
    Endfor
until all virtual nodes have been mapped
4. Output the mapping or failed results.

The failed conditions are not shown to increase the readability of the algorithm. When the unused neighboring node satisfying the constraints for one virtual node does not exist or when the inflation factor is violated, the algorithm is deemed to have failed.

4.3.3. Entire ACO algorithm for VNE problem

The entire ACO algorithm for the VNE problem includes four stages: the initial stage, the decomposition stage, the mapping stage for all sub-graphs (ring structures and tree structures), and the merging stage. An iteration process of the ACO algorithm includes the last three stages. Each ant decomposes the virtual request into a combination of ring structures and tree structures in the decomposition stage, maps these structures into the SN one by one using their individual algorithms in the mapping stage and merges the mapping results into an entire mapping in the substrate using the recorded associations in the merging stage. Some ants may obtain an entire mapping result. We evaluate the result using a fitness function and update the pheromone matrix according to some update rules. Many iteration processes are used to search for the optimal solution until the terminal condition of the ACO algorithm is reached.

1. The initial stage uses the pre-computation algorithm for the SN topology and the network topology of the current virtual request to find their ring characteristics.
2. The decomposition stage is used only for a virtual request. An ant begins at a randomly selected virtual node v whose corresponding pheromone value $\tau_{vj} > 0$ (which means that node v is in some ring structures) and searches for the ring structures in the VN topology. When a ring is found, it is recorded as a sub-graph and then pruned. The links that connect the ring and the residual graph are recorded as the associations and then pruned. The nodes, which are the two endpoints of these association links are labeled as leaders; the initial degrees are recorded as the current degrees for these leaders. The process is continued in this residual topology until no ring structures exist. The residual topologies are some tree structures. In this stage, different ants may obtain different decomposition results, and the difference makes the mapping process highly flexible, which results in a high acceptance ratio.
3. At the mapping stage, all the decomposed sub-graphs are plotted into the SN separately using the algorithms proposed in Sections 4.3.1 and 4.3.2. The ring structures are mapped first, followed by the tree structures. These structures are mapped one by one. When a structure is mapped, all the candidate and mapping nodes in the substrate are labeled as used and are not employed in other mappings. Through this method, we aim to maintain the topology invariance of the virtual request in the mapping process. A virtual node, which is a leader, is randomly selected when mapping a ring structure; this node serves as the start node in the mapping process. When mapping a tree structure, a leader connected with a mapped ring structure is randomly selected, and the associated leader in the ring structure is used as the root of this tree structure and as the starting element of the mapping process. Through this method, some associations are addressed while other associations might be excluded. These excluded associations are mapped into the SN in the merging stage.
4. At the merging stage, all the mapped substrate sub-graphs are combined into an entire graph by mapping the excluded associations. From the above mapping stage, some associations that connect a ring structure and a tree structure are mapped; however, some of these associations and others that connect two ring structures are still not mapped. Such associations are mapped in the merging stage. The key problem is searching for an appropriate mapping edge or path for a virtual link whose two endpoints are mapped. This problem can be addressed using a pheromone routing table, as described by Dorigo [15]. Based on this table, an unused node is selected one by one. A long path is selected in some cases. As the complexity of a virtual request influences the merging stage, this stage can be the most difficult part of a mapping process for a virtual request. In this study, we relax the topology invariance constraint in this stage. When searching for an edge or a path for the associations, some used nodes can be used again if unused nodes are unavailable. To overcome the influence of the relaxing method, we introduce a punish factor in the fitness function, as described in Section 3.4. We can maintain the degree of topology invariance by adjusting the parameters in Eq. (21). By giving some pheromone increases to the infeasible mappings, some mapped ring or tree structures in such mappings can obtain opportunities to be reused as starting points of searching process of an ant in the following iteration processes. A feasible mapping may be determined by exploiting the neighboring areas of these mapped structures or by simply merging mapped structures from different infeasible mappings. The relaxed method and penalty factor provide our algorithm with additional opportunities to determine feasible mappings, which may result in enhanced performance. The value of parameter Δ in Eq. (21) can determine the value of pheromone increase that will be set for these infeasible mappings, and a large value indicates a relaxed tendency to reuse an infeasible mapping.

The entire mapping algorithm for a general virtual request is presented as follows.

Input: A virtual request $G^R = (V^R, E^R)$, where each node or link has its resource requirement $C^{R,V}_{cpu}(v)$ or $C^{R,E}_{bw}(e)$;
A substrate network $G^S = (V^S, E^S)$, where each node or link has its resource $C^{S,V}_{cpu}(v)$ or $C^{S,E}_{bw}(e)$;
Output: $f : v \in V^R \mapsto u \in V^S$, $f : (v_{i-1}, v_i) \in E^R \mapsto (u_{i-1}, c_1, \ldots, c_k, u_i)$, or fails (reject the virtual request)
1. Initialization; (initial stage and the initialization of all parameters);
2. While (the terminal conditions of the ACO algorithm have not been reached)
For each ant
randomly decompose the virtual request, record the sub-structures (ring structures and tree structures) and the associations;
map the sub-structures one by one using their own mapping algorithms;
merge the mapping structures in the substrate by mapping the associations;
record the mapping results or failures.
endfor
evaluate the obtained mapping results by the fitness function;
update the pheromone values;
endwhile
3. Output the best mapping results or failures.

4.4. Pheromone update rules

When an iteration process finishes, the ants obtain a number of solutions. These solutions are evaluated by the fitness function defined in Eq. (21). The knowledge for finding improved solutions is extracted from these solutions and used in the next iteration process. The knowledge is stored in the pheromone matrix (ϕ_{ij}), and the element ϕ_{ij} indicates that the expectation of the virtual node i is mapped to the substrate node j to obtain the optimal solution. After one iteration process, the pheromone matrix is updated according to the solutions obtained by the ants. We use three pheromone update rules. The first rule is that the elements of the best solution should result in a pheromone increase. The second rule is that all the pheromones should evaporate to avoid excessive pheromone accumulation. The third rule is that pheromone normalization reflects the relative probability. Thus, the pheromone update rules are defined by the following formula:

$$\phi_{ij} = (1 - \lambda)\phi_{ij} + f(s^*) \times x_{ij} \tag{26}$$

where s^* is the best valid solution, λ is the evaporation rate of the pheromone, and

$$x_{ij} = \begin{cases} 1, & \text{if } \ f : i \mapsto j \\ 0, & \text{else} \end{cases} \tag{27}$$

are binary variables, which indicate the elements of the best solution. At the same time,

$$\sum_j \phi_{ij} = 1, \quad \forall i \tag{28}$$

guarantees the normalization of the pheromone.

5. Simulation results and analysis

The simulation environment, the performance metrics, and the simulation results are discussed in this section.

5.1. Simulation environment

The simulation environment is a modified version of the simulation environment used in [10]. We inherit the realization of the GN-SP, and ViNE-LB algorithms in their simulation platform [37]. Several modifications are made. First, to compare optimization performances fairly, the location information in their algorithm is deleted, which allows a VN request to be embedded into any suitable part of the SN. Second, hop count is used as the metric to compute the shortest path by setting the weight of each link as 1 in their algorithms. Third, to increase randomization in their algorithm, deterministic and random node selection is applied with equal probability. Thus, the name of the compared algorithm is ViNE-LB. Fourth, some synthesized topologies whose construction methods are similar with those presented in [10] and a practical network topology extracted by Rocketfuel [38] are used. Lastly, we modify and introduce several performance metrics to reflect the node and link utilization and stress.

The synthesized SN topologies used in our algorithm are generated by the Georgia Tech Internetwork Topology Models (GT-ITM) tool [39] with 50 nodes in a 25 × 25 grid. Each pair of SN nodes is connected randomly with a probability of 0.5 or 0.8 in different SN topologies to reflect the density of a network topology. The practical SN topology used is the Sprint network topology which is extracted by Rocketfuel. We only extract the backbone nodes and the links between them to form our SN network which has 377 nodes and 1450 edges (that is, some difference with [38]). The CPU capacity value of each SN node and the bandwidth value of each SN link are uniformly distributed between 50 and 100. The VN request topologies are also generated by the same tool with 2 to 10 nodes in the same grid. Each pair of VN nodes is connected randomly with a probability of 0.5. The CPU and bandwidth requirements of the VN nodes and VN links are uniformly distributed from 0 to 20 and from 0 to 50, respectively.

The VN requests arrive according to a Poisson process. We adjust the average rate from 4 to 8 each 100 time units to change the numbers of VN requests in the network for the synthesized SN topologies. The average rate is 40 for each 100 time units for the practical Sprint topology. The lifespan of a VN request is exponentially distributed, and the average is 1000 time units. The simulation time is fixed at 50,000 time units.

Our proposed algorithm is called ACO-TD (ACO algorithm based on Topology Decomposition). To evaluate the performance of our proposed algorithm, two algorithms are used as counterparts: ViNE-LB [10]: deterministic or randomized node mapping with MCF link mapping; GN-SP [6]: greedy node mapping with shortest path link mapping.

We realize ACO-TD uniquely in a PC. Our proposed algorithm is run on Ubuntu 12.04, and coded by C++. Moreover, the compiler is g++ 4.6. We use the same files of SN topologies, and VN requests as input for our algorithm and the compared algorithms. Then we run them separately, output the results to corresponding files, and compare their performances based on the output files.

5.2. Performance metrics

We define the following performance metrics to examine the performance of our algorithm and the two algorithms used for comparison.

(1) **Acceptance ratio**: Measures the percentage of total VN requests accepted by an algorithm over a given period [10].
(2) **Average generated revenue:** Measures the generated revenue of an embedding algorithm over time, which is defined according to [10,20,22].
(3) **Provision cost**: Measures the provisioning cost of embedding a VN request (defined in Eq. (2)) [10].

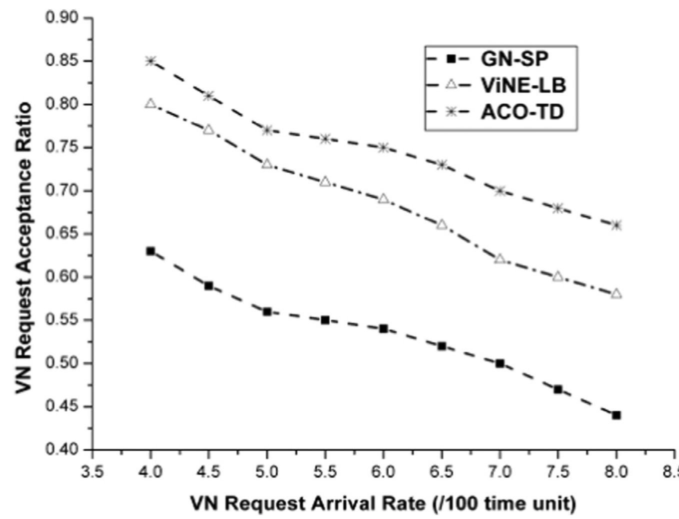

Fig. 2. VN request acceptance ratio with different VN request arrival ratios.

(4) **Node (link) utilization:** Measures the CPU (bandwidth) usage percentage of an SN node (link) when an algorithm embeds some VN requests in the SN [10].
(5) **Standard deviation of node (link) utilization:** Evaluates the characteristics of the load balancing of an embedding algorithm. These load balancing factors can reflect the fluctuation degree of the resource usage of nodes and links. A low value of this metric equates to great balancing, which also indicates that a large number of VN requests can be accepted and that a high revenue can be obtained [4].
(6) **Average node (link) stress:** Defines the average times a VN request uses a node or a link [6]. To maintain the characteristics of the topology invariance of a VN request, the value of our proposed algorithm is always 1.0. This value is larger than 1 for the other algorithms. This condition might be the cause of the low utilization of some nodes or links.

5.3. Simulation results

In the following simulation, the edge inflation factor is set to $\delta \leq 3$, which means that a virtual link is restricted to be mapped to a substrate path whose hop length is no longer than 3. The reason for this decision is that in our simulation environments larger inflation factor cannot help the algorithm obtain better performance obviously.

5.3.1. Simulation results on the synthesized SN topologies

In this section, the synthesized SN topologies with 50 nodes are used in the simulation.

Fig. 2 shows the acceptance ratio of the three algorithms when the average number of VN requests changes from 4 to 8 each 100 time units. The tendency of the acceptance ratio is lower with a large number of VN requests because the SN has resource constraints. When too much virtual requests require mapping to the SN, a large number of requests are rejected. The figure also shows that our algorithm has a higher acceptance ratio than the other algorithms. Therefore, the proposed algorithm allocates resources more efficiently and accepts more requests compared with the other algorithms. This finding is attributed to the random but shortest paths used, which increases the flexibility of the mapping process, and to the topology invariance, which avoids the repeatable use of some substrate resources as hot spots. This condition favors the mapping of other requests.

Fig. 3 shows the average generated revenue by the accepted VN requests over the simulation period. The average generated revenue tends to be high given a large number of VN requests. This condition

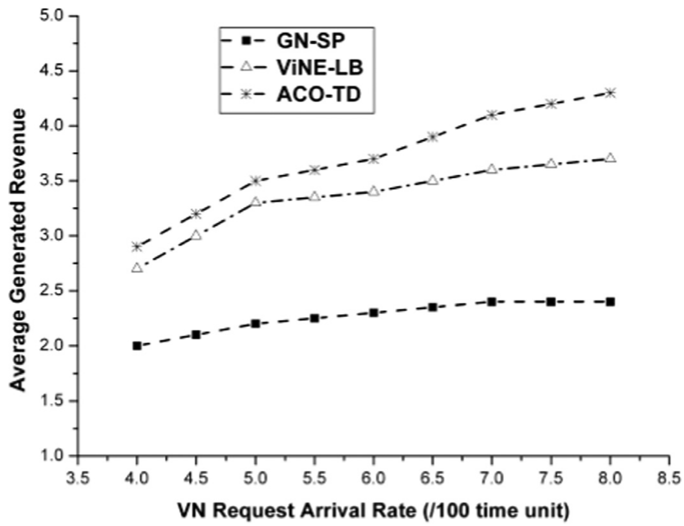

Fig. 3. Average generated revenue with different VN request arrival ratios.

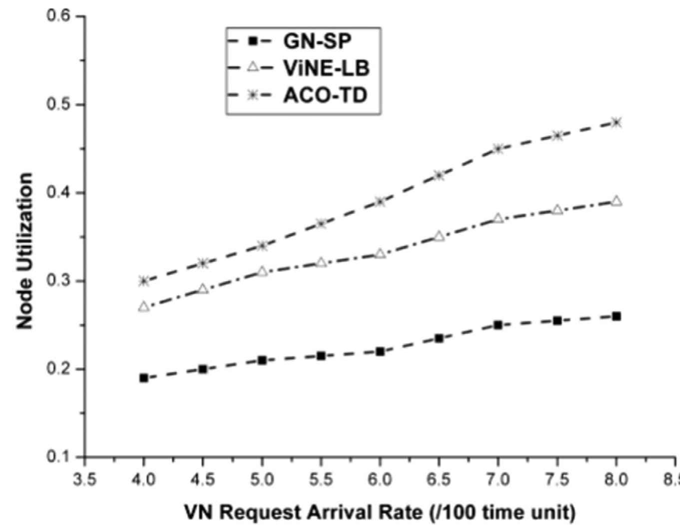

Fig. 5. Node utilization of the SN with different VN request arrival ratios.

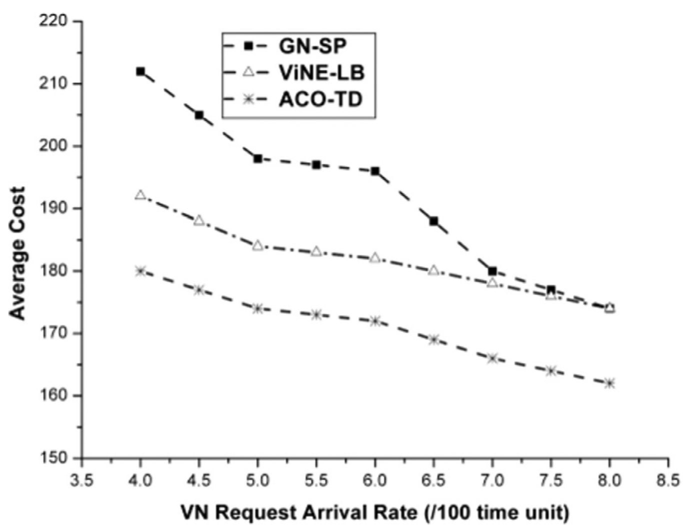

Fig. 4. Average cost of a VN request with different VN request arrival ratios.

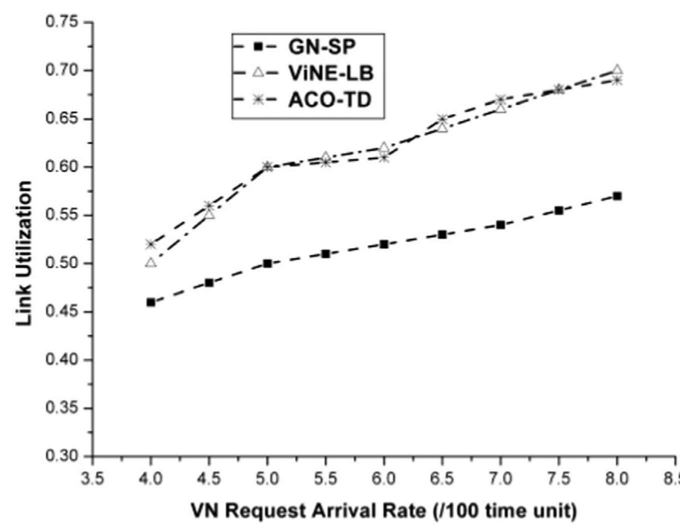

Fig. 6. Link utilization of the SN with different VN request arrival ratios.

is true because several appropriate requests will be mapped given the generation in certain ranges of a large number of VN requests whose requirements are randomly uniform. A high generated revenue, along with a high acceptance ratio, indicates an enhanced performance. The tendency of the generated revenue to increase with the number of requests is more evident in our algorithm than in the other algorithms. Such result, along with the high acceptance ratio, shows the efficient use of the resources of the SN through efficiently used substrate nodes and links.

Fig. 4 shows the average provision cost of a VN request. The cost decreases as the number of VN requests increases because many opportunities for appropriately mapping to the SN become available as the number of requests generated in some ranges increases. The GN-SP algorithm, greedy node selection followed by shortest path selection, causes some SN nodes and links disconnected rapidly and makes the following requests difficult to find a mapping of less cost. The ViNE-LB algorithm, splittable flow along with load balance, uses some longer paths and seperates one virtual link onto several SN paths to obtain better performance than the GN-SP algorithm. However, the idea of splittable flow also breaks the SN resources into fragments. Our algorithm gains less provision cost compared with the other algorithms because the proposed algorithm decomposes the whole VN request into ring structures and tree structures and maps them using different algorithms based on the characteristics of topology invariance. This mapping process decreases the interference between the two types of structures and results in a highly detailed mapping. Another reason for our algorithm obtaining better performance relies on the evolutionary iteration processes of the ACO algorithm.

Figs. 5 and 6 show the node utilization and link utilization, respectively. The proposed algorithm shows higher node utilization than the other algorithms and a link utilization similar to that of ViNE-LB. This finding indicates that the topology invariance of our algorithm can improve the utilization of substrate nodes and offers other opportunities for mapping subsequent VN requests. The low node utilizations of the compared algorithms and their link utilizations similar to that of the proposed algorithm indicate that some substrate links are used several times when mapping a VN request. The sharing of the same links not only violates the topology invariance of the VN request, but also makes some nodes connected with these links have less opportunities to be used. In this case, isolating the SN and influencing the mapping of the next VN requests become highly likely.

Figs. 7 and 8 respectively show the standard deviation of the node utilization and link utilization of the SN. These definitions can describe the load balance of the nodes and links. The standard deviation of the node utilization of our algorithm is lower than that of the other algorithms. Furthermore, the standard deviation of the link

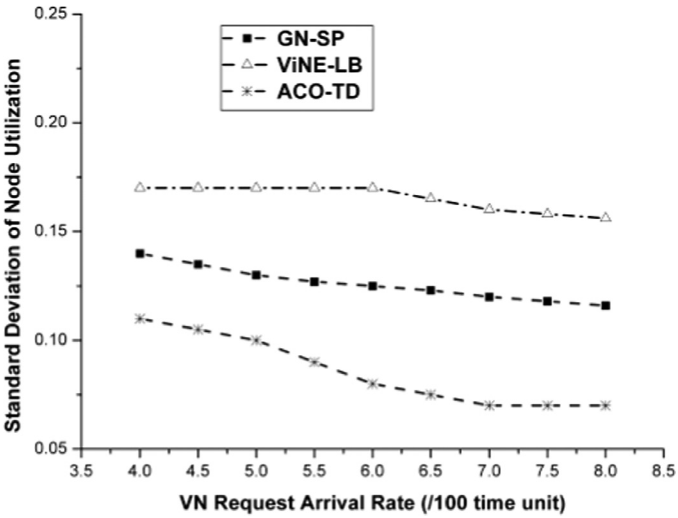

Fig. 7. Standard deviation of node utilization with different VN request arrival ratios.

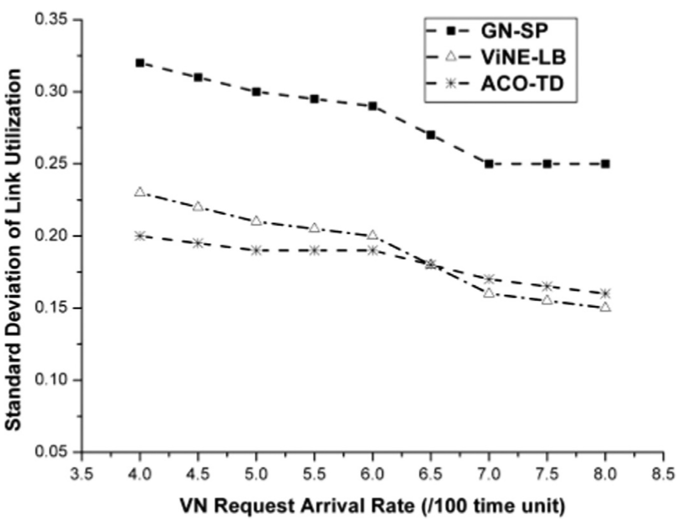

Fig. 8. Standard deviation of link utilization with different VN request arrival ratios.

utilization of our algorithm is similar to that of the other algorithms. This finding indicates that compared with the other algorithms, our algorithm can use substrate nodes more efficiently and achieve a higher acceptance ratio and larger generated revenue. In addition, the proposed algorithm maps the VN requests more flexibly and efficiently compared with the other algorithms by analyzing the topology characteristics of the SN and VN requests.

We define the average node stress and link stress of a VN request to reflect the topology invariance of the algorithms. For the GN-SP and ACO-TD algorithms, these metrics are easy to compute because a single path is used for each virtual link. For the ViNE-LB algorithm, we only count the shared nodes and links among different virtual links, whereas the nodes and edges shared by multiple paths for a virtual link are disregarded because we mainly focus on topology invariance in this study. Figs. 9 and 10 show the results. Our algorithm has the constant value of 1.0, whereas the other algorithms have a larger value. This condition indicates that some nodes and links are used relatively frequently when mapping a VN request. The analysis of the SN and the VN requests used in our algorithm provides a unique opportunity to improve the performance of the mapping process. The ACO algorithms for ring structures and tree structures can take advantage of this improved mapping process through randomization and evolution.

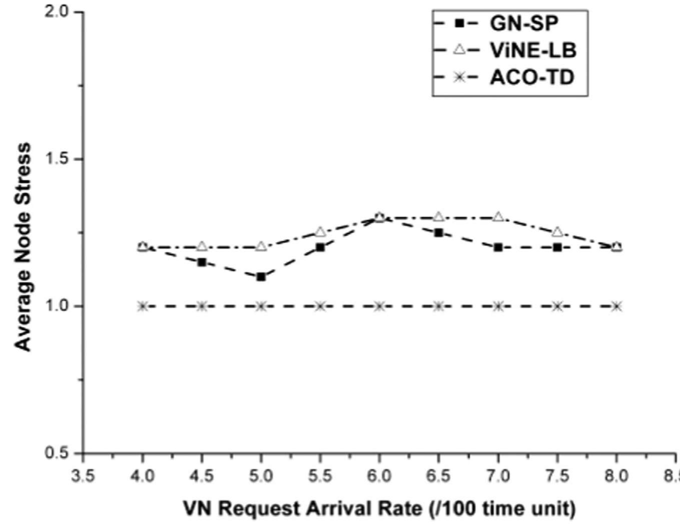

Fig. 9. Average node stress with different VN request arrival ratios.

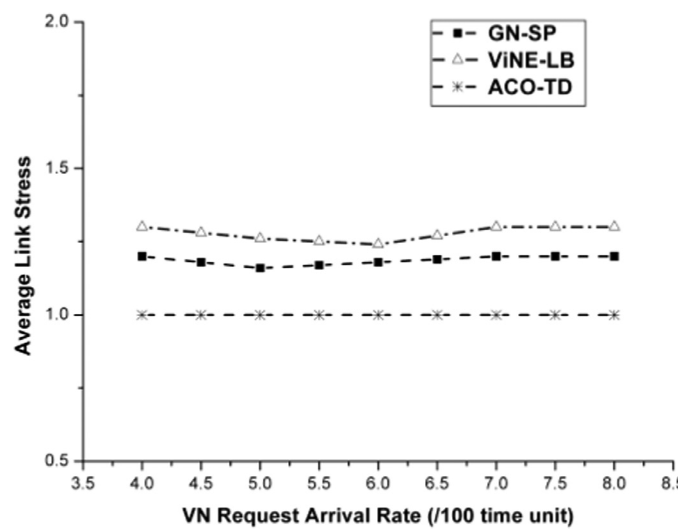

Fig. 10. Average link stress with different VN request arrival ratios.

Table 1
The evolutionary capacity of ACO-TD.

Iteration times	Acceptance ratio	Revenue	Cost
1	0.502	1.653	219.087
2	0.613	1.825	206.435
10	0.847	2.812	184.276

Table 2
Effectiveness of the components of fitness function.

	Acceptance ratio	Revenue	Cost
$\alpha = 0$	0.751	2.134	203.125
$\beta = 0$	0.803	2.512	192.381
$\Delta = 0$	0.826	2.724	186.253

The optimization results of one, two, and ten iteration processes are shown in Table 1 to illustrate the evolutionary capacity of the ACO-TD algorithm. The generation rate of VN requests is set to 4 each 100 time units. The results indicate that performance can be enhanced through multiple randomization and pheromone-based iterations.

Table 3
Comparative performance on hub-and-spoke topologies.

	Acceptance ratio	Revenue	Cost
GN-SP	0.617	1.932	198.103
ViNE-LB	0.802	2.751	176.213
ACO-TD	0.819	2.809	183.516

Table 4
Comparative performance on mesh topologies.

		Acceptance ratio	Revenue	Cost
$cp = 0.5$	SN-SP	0.492	1.657	232.825
	ViNE-LB	0.612	2.482	225.165
	ACO-TD	0.407	1.513	217.324
$cp = 0.8$	SN-SP	0.514	1.741	203.143
	ViNE-LB	0.637	2.592	218.954
	ACO-TD	0.726	2.967	193.251

Table 5
Comparative performance on sprint.

	Acceptance ratio	Revenue	Cost
GN-SP	0.748	22.745	201.728
ViNE-LB	0.826	29.385	182.465
ACO-TD	0.867	31.946	170.946

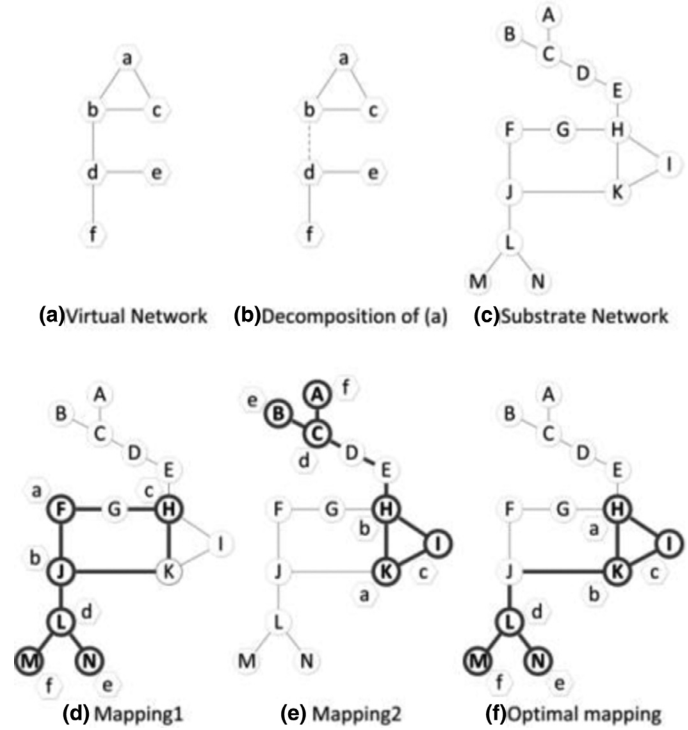

Fig. 11. Evolutionary iteration process of ACO-TD.

Table 2 shows the effectiveness of different components in the fitness function (which is defined in Eq. (21)). $\alpha = 0$, $\beta = 0$, or $\Delta = 0$ indicates that the cost factor, load balance factor, or penalty factor, respectively, is not used to evaluate a mapping solution. The generation rate of the VN requests is set to 4 each 100 time units. The simulation results indicate that the cost factor has the most influence on the optimization performance, whereas the penalty factor has the least contribution.

To illustrate the performance of ACO-TD and its counterpart algorithms on specific VN requests, two simulations are conducted, and the generation rate of VN requests is set to 4 each 100 time units. Table 3 presents the simulation results on hub-and-spoke VN topologies, which exhibit the same tendencies. Table 4 presents the results on the fully meshed VN requests. In this simulation, we use two SN topologies, and set the connection probabilities cp between any two nodes to 0.5 and 0.8, which present a regular random topology ($cp = 0.5$) and a denser topology ($cp = 0.8$). When the SN with $cp = 0.5$ is used, the ViNE-LB algorithm outperforms the other algorithms, and the ACO-TD algorithm exhibits the worst performance. This finding is attributed to the ACO-TD algorithm keeping the topology invariance, thus the nodes and edges in the SN which are not in any ring cannot be used. When the SN with $cp = 0.8$ is used, the ACO-TD algorithm exhibit the best performance, because numerous rings exist in a dense network.

5.3.2. Simulation results on the practical sprint network topology

In this section, the practical sprint network topology, which is extracted by Rocketfuel, is used as the SN. We only use the backbone nodes and the edges connected to them. Given that the network is significantly larger than the previous synthesized SN, the generation rate of the VN requests is set to 40 each 100 time units to generate sufficient VN requests.

Table 5 shows the simulation results of this SN. Although the values change, the same tendency remains.

5.4. Explanation of some ACO-TD components

In this section, we explain some components of our ACO-TD algorithm. We particularly focus on the benefit of the evolutionary iteration process of ACO to improve performance, the benefit of randomization in the mapping process of a tree structure, and the achievement of topology invariance. Finally, we present the parameters selection strategy. For clarity, the resource requirements of VN requests and SN resources are disregarded if they are not related to our discussion.

A. *The evolutionary iteration processes and decomposition may exhibit enhanced performance.* In Fig. 11b, the VN request is divided into a ring structure and a tree structure. During an iteration process, some ants search for mappings for the ring and tree structures. One ant may find a mapping shown in Fig. 11d, and the provision cost is 8 (for brevity, we use hop counts as the cost). Another ant may find another mapping shown in Fig. 11e, and the provision cost is also 8. The two mappings are not optimal. However, the mapped partitions in these two mappings will gain some pheromone increase, which indicates that they will be searched at a high probability in the following iteration. With the help of randomization and pheromone, in the following iteration process an ant may find an optimal mapping shown in Fig. 11f, and the provision cost is 7. Thus, the iteration process based on randomization and pheromone update is evolutionary and can help identify effective solutions.

B. *Random selection of sub-tree and root may increase the flexibility of mapping process of a tree structure.* We randomly select the root of a tree or a sub-tree to provide multiple starting points, which result in different search sequences. Meanwhile, we map a sub-tree in one stage to search an area (a block with some nodes and their links) simultaneously. Fig. 12 provides an example to illustrate the benefit of our algorithm. When an ant maps the VN with a BFS (broad first search) sequence and a greedy resource idea, the mapping fails, as shown in Fig. 12c. If another ant considers sub-tree mapping and performs random selection, it will obtain a feasible mapping shown in Fig. 12d. With multiple ants that start at a stochastic node and conduct a random search sequence, the probability to obtain a feasible solution is high. A tree structure has increased chances of being accepted by obtaining a feasible mapping.

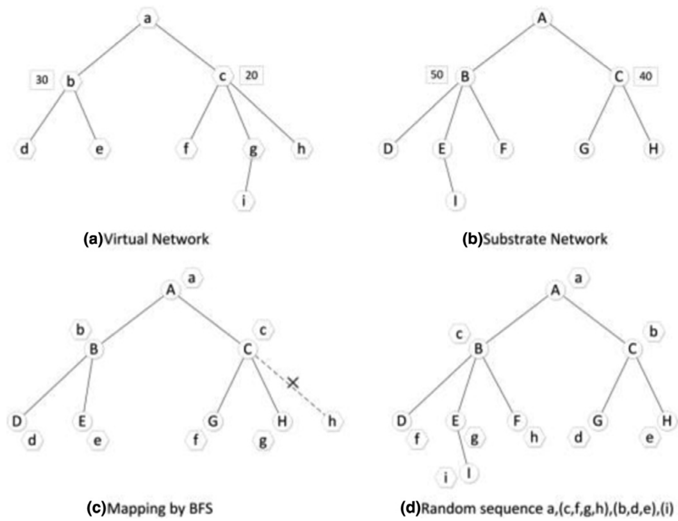

Fig. 12. Random sequence for tree mapping.

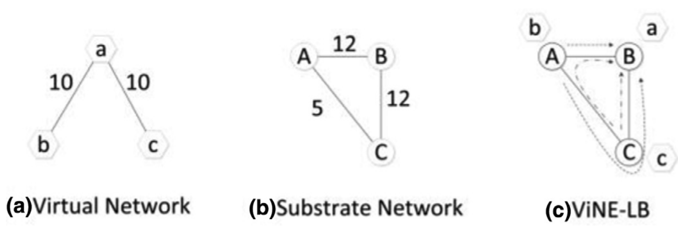

Fig. 13. Benefit of topology invariance.

C. *Benefit of VN topology invariance to resource management and protocol deployment in an SN.* In addition to the aforementioned reliability consideration for topology invariance, resource management and protocol deployment are also problems in an SN. If an SN supports the splitting data transmission, the mapping result by ViNE-LB algorithm is shown in Fig. 13c. When this mapping is realized in the SN, numerous states are eventually employed in some nodes when a connection-oriented protocol is used (e.g., MPLS) for multiple paths established for a virtual link, or loops will occur when a connectionless protocol is used (e.g., IP). In the figure, a loop exists between SN node B and C. These will translate more complicated schemes for an SN to avoid state scalability problem or loop-free mechanisms. Loops may also occur in an SN in other algorithms that use the shortest path for a virtual link when there are shared mapped nodes or links between any two virtual links.

D. *Selection of parameters.* The optimal selection of parameters in each ACO algorithm is another NP-hard problem. Different combinations of parameter values significantly influence the exploration and exploitation behavior of ants, and eventually, optimization performance [40]. In our pervious works [41], we model the optimal parameter selection problem as determining an optimal point in multi-dimensional space, and present an algorithm based on PSO to gain a near optimal combination of parameter values. These values depend on the instances of an optimization problem. Thus, the parameter selection algorithm is used as a part of our ACO algorithm.

6. Conclusion

We modeled the VNE problem based on virtual network topology invariance to guarantee the reliability requirements of SPs. We analyzed the differences between ring structure and tree structure mapping process and proposed a modified ACO algorithm for the VNE problem based on graph decomposition. We also obtained the ring characteristics of the SN and VN requests through pre-computation and decomposed each VN request into a combination of ring structures and tree structures by using pre-computation results. We designed different mapping algorithms for the ring and tree structures. The ring structure mapping algorithm uses the ring characteristic of the SN, and the tree structure mapping algorithm uses a tree-indexed random walking construction process. These algorithms can use the topology characteristics efficiently for an enhanced performance.

To satisfy the requirement of the VN in the VN topology, we proposed the term "topology invariance," which avoids the unnecessary sharing of nodes and links. Candidate nodes were used to achieve topology invariance. The proposed algorithm not only guarantees topology invariance but also improves the load balance of the SN to accept several future VN requests.

Acknowledgments

This study is supported by the Natural Science Foundation of Shandong Province (Grant No. ZR2013FM029) and the University Innovation Project of Jinan (Grant No. 201303010). Special acknowledgment is extended to Dr. Mosharaf Chowdhury, who provided his simulation platform.

References

[1] T. Anderson, L. Peterson, S. Shenker, J. Turner, Overcoming the internet impasse through virtualization, Computer 38 (2005) 34–41.
[2] M. Chowdhury, R. Boutaba, A survey of network virtualization, Comput. Netw. 54 (2010) 862–876.
[3] D. Anderson, Theoretical approaches to node assignment, Unpublished Manuscript, 2002. http://www.cs.cmu.edu/~dga/papers/andersen-assign.ps.
[4] M. Yu, Y. Yi, J. Rexford, M. Chiang, Rethinking virtual network embedding: substrate support for path splitting and migration, ACM SIGCOMM Comput. Commun. Rev. 38 (2008) 17–29.
[5] A. Fischer, J.F. Botero, M.T. Beck, H. Meer, X. Hesselbach, Virtual network embedding: a survey, IEEE Commun. Surveys Tutorials 15 (4) (2013) 1888–1906.
[6] Y. Zhu, M.H. Ammar, Algorithms for assigning substrate network resources to virtual network components, in: Proceedings of INFOCOM 2006, Barcelona, Spain, 2006, pp. 1–12.
[7] X. Cheng, S. Su, Virtual network embedding through topology awareness and optimization, Comput. Netw. 56 (2012) 1797–1813.
[8] X.L. Li, H.M. Wang, B. Ding, X.Y. Li, D.W. Feng, Resource allocation with multi-factor node ranking in data center networks, Future Generation Comput. Syst. 32 (2014) 1–12.
[9] H.Y. Cui, S.H. Tang, J.Y. Chen, Y.J. Liu, A novel method of virtual network embedding based on topology convergence-degree, in: Proceedings of IEEE ICC, 2013, pp. 246–250.
[10] M. Chowdhury, M.R. Rahman, R. Boutaba, ViNEYard: virtual network embedding algorithms with coordinated node and link mapping, IEEE/ACM Trans. Network. 20 (2012) 206–219.
[11] I. Fajjari, N. Aitsaadi, G. Pujolle, H. Zimmermann, VNE-AC: virtual network embedding algorithm based on ant colony metaheuristic, in: Proceedings of IEEE ICC, Kyoto, Japan, 2011, pp. 1–6.
[12] J. Lischka, H. Karl, A virtual network mapping algorithm based on subgraph isomorphism detection, in: Proceedings of VISA'09, Barcelona, Spain, 2009, pp. 81–88.
[13] S. Qing, J. Liao, J. Wang, X. Zhu, Q. Qi, Hybrid virtual network embedding with K-core decomposition and time-oriented priority, in: Proceedings of IEEE ICC, Ottawa, Canada, 2012, pp. 2695–2699.
[14] T. Huang, J. Liu, J.Y. Chen, Y.J. Liu, A topology-cognitive algorithm framework for virtual network embedding problem, China Commun. 11 (4) (2014) 73–84.
[15] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst., Man, Sybern. Part B 26 (1996) 29–42.
[16] M. Dorigo, T. Stützle, Ant colony optimization: overview and recent advances, Handbook of Metaheuristics 146 (2010) 227–263.
[17] Q. Zhu, H.Q. Wang, H.W. Lv, Z.D. Wang, VNE-AFS: virtual network embedding based on artificial fish swarm, J. Commun. 33 (2012) 170–177.
[18] Z. Zhang, X. Cheng, S. Su, Y. Wang, K. Shuang, Y. Lou, A unified enhanced particle swarm optimization-based virtual network embedding algorithm, Int. J. Commun. Syst. 26 (8) (2013) 1054–1073.
[19] X.L. Chang, X.M. Mi, J.K. Muppala, Performance evaluation of artificial intelligence algorithms for virtual network embedding, Eng. Appl. Artif. Intell. 26 (2013) 2540–2550.
[20] J.F. Botero, M. Molina, X.H. Serra, J.R. Amazonas, A novel paths algebra-based strategy to flexibly solve the link mapping stage of VNE problems, J. Netw.Comput. Appl. 36 (2013) 1735–1752.
[21] G. Even, M. Medina, G. Schaffrath, S. Schmid, Competitive and deterministic embeddings of virtual networks, Theoretical Comput. Sci. 496 (2013) 184–194.

[22] G. Sun, H.F. Yu, V. Anand, L.M. Li, A cost eficient framework and algorithm for embedding dynamic virtual network requests, Future Generation Comput. Syst. 29 (2013) 1265–1277.

[23] I. Houidi, W. Louati, D. Zeghlache, A distributed virtual network mapping algorithm, in: Proceedings of IEEE ICC, 2008, pp. 5634–5640.

[24] M.T. Beck, A. Fischer, H. Meer, J.F. Betero, X. Hesselbach, A distributed, parallel, and generic virtual network embedding framework, in: Proceedings of IEEE ICC, 2013, pp. 3471–3475.

[25] M. Demirci, M. Ammar, Design and analysis of techniques for mapping virtual networks to software-defined network substrates, Comput. Commun. 45 (2014) 1–10.

[26] M. Chowdhury, F. Samuel, R. Boutaba, Polyvine: policy-based virtual network embedding across multiple domains, in: Proceedings of ACM SIGCOMM VISA, 2010, pp. 49–56.

[27] C. Werle, R. Bless, P. Papadimitriou, I. Houidi, W. Louati, D. Zeghlache, L. Mathy, Building virtual networks across multiple domains, in: Proceedings of ACM SIGCOMM, 2011, pp. 412–413.

[28] W. Yeow, C. Westphal, U.C. Kozat, Designing and embedding reliable virtual infrastructures, ACM SIGCOMM Comput. Commun. Rev. 41 (2) (2011) 57–64.

[29] M.R. Rahman, R. Boutaba, SVNE: survivable virtual network embedding algorithms for network virtualization, IEEE Trans. Netw. Service Manage. 10 (2) (2013) 105–118.

[30] J.F. Botero, X. Hesselbach, Greener networking in a network virtualization environment, Comput. Netw. 57 (2013) 2021–2039.

[31] S. Su, Z. Zhang, A.X. Liu, X. Cheng, Y. Wang, X. Zhao, Energy-aware virtual network embedding, IEEE/ACM Trans. Network. 22 (5) (2014) 1607–1620.

[32] S. Shakya, N. Pradhan, X. Cao, Z. Ye, C. Qiao, Virtual network embedding and reconfiguration in elastic optical network, in: *Proceedings of Globecom*, 2014, pp. 2160–2165.

[33] L. Xu, G. Tan, X. Zhang, Probabilistic inference-based service level objective-sensitive virtual network embedding, Comput. Commun. 57 (2015) 25–36.

[34] C. Liang, F.R. Yu, Wireless network virtualization: a survey, some research issues and challenges, IEEE Commun. Surveys Tutorials 17 (1) (2015) 358–380.

[35] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP Completeness, W. H. Freeman & Co Ltd., 1978.

[36] L.P. Cordella, P. Foggia, C. Sansone, M. Vento, A (sub)graph isomorphism algorithm for matching large graphs, IEEE Trans. Pattern Anal. Mach. Intell. 26 (2004) 1367–1372.

[37] http://mosharaf.com/TONSim-0.tar.gz. Accessed May 15, 2013.

[38] N. Spring, R. Mahajan, D. Wetherall, Measuring ISP topologies with rocketfuel, in: Proceedings of ACM SIGCOMM, 2002, pp. 133–145.

[39] E. Zegura, K. Calvert, S. Bhattacharjee, How to model the Internetwork, in: Proceedings of IEEE INFOCOM, 1996, pp. 594–602.

[40] M. Dorigo, T. Stutzle, Ant Colony Optimization, Tsinghua University Press, 2007.

[41] H. Xia, H. Wang, X. Chen, A kind of ant colony parameter adaptive optimization algorithm based on particle swarm optimization thought, J. Shandong Univ. (Eng. Sci.) 40 (3) (2010) 26–30.