2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
Banff Center, Banff, Canada, October 5-8, 2017

An Ant Colony System Based Virtual Network Embedding Algorithm

Jia-Bin Wang
Sun Yat-sen University
Guangzhou, China

Wei-Neng Chen, Hao Cong, Zhi-Hui Zhan, Jun Zhang
South China University of Technology
Guangzhou, China
cwnraul634@aliyun.com

***Abstract*—The virtual networking embedding (VNE) problem is a core issue in network virtualization. This is also a challenging problem as it contains different kinds of constraints, and its complexity becomes even higher in an online VNE problem with thousands of virtual network (VN) requests. In this paper, we proposed an ant colony system based VNE algorithm, called ACS-VNE, for the online VNE problem. The benefits of ACS-VNE are threefold. First, it is an ACS based algorithm so it can take full advantage of the dynamically changing heuristic information and pheromone to improve the quality of a solution. Second, different from previous work that only considers the resource of nodes in node mapping phase, we take the distance message related to links into consideration so that we can reduce the cost of VN requests. The last but not least, the algorithm tries to reduce the cost for every single VN and it helps to increase the possibility of accepting more future VN requests. The proposed method is tested on both the single VN request VNE problem and the online VNE problem. Experimental results show that the proposed algorithm outperforms some previous approaches in terms of average revenue and acceptance ratio, and the results also have a relatively low cost.**

Keywords—virtual networking embedding (VNE), ant colony system (ACS), ant colony optimization (ACO) network virtualization

I. Introduction

The Internet has been improved rapidly in recent decades. But it is difficult for the existing network architecture to satisfy new applications that spring up one after another, and this results in Internet impasse problems. Network virtualization [1] allows multiple heterogeneous virtual networks to coexist on a shared infrastructure so that it can overcome the resistance of the current Internet to architecture change. In a network virtualization environment, multiple service providers (SPs) provide service for customers by creating heterogeneous virtual networks (VNs). All these VNs are embedded in the substrate networks (SNs) possessed by the infrastructure providers (InPs). And then SPs need to pay InPs. Usually, the main purpose of InPs is increasing revenue and reducing cost.

In network virtualization, the core issue is virtual network embedding (VNE). VNE aims to map VN requests on one or more SNs, and there are two phases of mapping, node mapping and link mapping. Node mapping is NP-hard [2], and the link mapping procedure after node mapping is also NP-hard [2]. In order to reduce the complexity of VNE, many previous studies neglect some important constraints [3, 4] or only consider the offline VNE problem [3, 5]. Chowdhury et al. [6] model a VNE problem as a mixed integer programming (MIP) and their algorithm can solve the online VNE problem with kinds of constraints. However, the algorithm is time consuming on a large-scale problem. In this paper, we use an integer liner programming (ILP) proposed by [2].

Since ILP is a NP-hard problem, traditional mathematical approaches such as branch and bound and cutting plane cannot solve problems with large scale. So we are looking for help from metaheuristic techniques. Metaheuristic algorithms such as genetic algorithm (GA) [7], simulated annealing (SA) [8], particle swarm optimization (PSO) [9] and ant colony optimization (ACO) [10, 11, 12] have been improved efficient in NP-hard problems and the improved algorithms of them can solve large-scale problems efficiently. The authors in [2] try to solve the ILP using a unified enhanced PSO-based algorithm, but they only consider the resource of substrate nodes in node mapping. And some ACO based algorithms [13, 14] do not make full use of the knowledge of SN in node mapping.

ACO is a robust algorithm and suitable for complex practical problems [15, 16, 17] such as the traveling salesman problem (TSP). In ACO, each ant can build a solution by waking on the searching space. Every ant is guided by two factors: pheromone and heuristic information. Pheromone records the searching experience of ants, while heuristic information is problem-specific knowledge. The performance of ACO is significantly influence by heuristic information and a good design of heuristic information can make the best use of the knowledge of the problem. In VNE, if we design the heuristic information by using several kinds of message of SN, we can get a near optimal solution fast. Therefore, ACO is suitable for the VNE problem. However, the basic ACO has a slow convergence and the solution of basic ACO is likely to be trapped in local optimal. So we need an improved ACO algorithm to be applied on the VNE problem.

In response to all the deficiencies above, this paper proposed a novel algorithm, the ant colony system based virtual network embedding algorithm, referred as VNE-ACS

This work was supported in part by the National Natural Science Foundation of China under Grant 61622206, Grant 61379061, and Grant 61332002, the Natural Science Foundation of Guangdong under Grant 2015A030306024, the Guangdong Special Support Program No. 2014TQ01X550, and the Guangzhou Pearl River New Star of Science and Technology No. 201506010002. *(Corresponding Author: Wei-Neng Chen).*

for the VNE problem. Ant colony system (ACS) [18] is an ACO-based algorithm and it has all the advantages of ACO. Furthermore, ACS has a unique state transition rule which can accelerate the convergence. To improve search diversity, a local pheromone update rule in some improved ACO algorithms is also used in ACS in case the solution stuck in a local optimal value. And the contributions we make to a revenue-aware VNE problem are as follows:

- We take the distance message of links as a part of heuristic information in node mapping stage and the results show this can cause less resource cost to the substrate network.
- We aim to minimize the resource cost of every single VN request so that we can leave more resource for SN in order to increase the possibility of accepting more future VN requests.

The experimental results show that the proposed algorithm performs significantly better than some other algorithms in terms of average revenue, acceptance ratio and the ratio of revenue to cost for the online VNE problem.

The rest of this paper is organized as follows. Section Ⅱ describes the network model and problem descriptions. Section Ⅲ introduces the proposed algorithm VNE-ACS. Section Ⅳ presents the experimental results. Section Ⅴ concludes this paper and gives an outlook of our future work.

II. NETWORK MODEL AND PROBLEM DESCRIPTION

In this section, we first give the network models for substrate networks and virtual networks. Then we give a brief description of the VNE problem. In the last three subsections, we describe more details of the VNE problem according to the algorithm we proposed in this paper. TABLE Ⅰ shows the notations and their descriptions we used in this section.

TABLE I. NOTATIONS AND THE DESCRIPTIONS

Notation	Description
n_S, n_i, n_j	Substrate nodes.
n_V, n_x, n_y	Virtual nodes.
l_S, l_{ij}	Substrate links, l_{ij} is the specific link between node n_i and n_j.
l_V, l_{xy}	Virtual links, l_{xy} is the specific link between node n_x and n_y.
p_S	A path in substrate network, it contains one or more links.
$ne(n_V,n_S)$	Its value is 1 if n_V is mapped to n_S, and 0 otherwise.
$le(l_V,l_S)$	Its value is 1 if l_V is routed on l_S, and 0 otherwise.
$CPU(n)$	The CPU capacity of a node n, n can be substrate or virtual.
$UC(n_S)$	The used CPU capacity of substrate node n_S.
$RC(n_S)$	The remaining CPU capacity of substrate node n_S.
$BW(l)$	The bandwidth of link l, l can be substrate or virtual.
$RB(l_S)$	The remaining bandwidth of substrate link l_S.

A. *Network Model*

We denote the substrate network as a weighted undirected graph $G_S = (N_S, L_S)$, where N_S is the set of substrate nodes and L_S is the set of substrate links. Every substrate node $n_S \in N_S$ has attributes such as memory storage, CPU capacity and location. In this paper, we only take CPU capacity into consideration, and $RC(n_S) = CPU(n_S) - UC(n_S)$.

The attribute of each substrate link $l_S \in L_S$ is the bandwidth between two substrate nodes and $RB(l_S)=BW(l_S)-UB(l_S)$. Furthermore, we denote the set of all the paths in the substrate network as P_S. Every $p_S \in P_S$ contains one or more links, so $RB(P_S) = \min\{RB(l_S) \mid l_S \in P_S\}$.

Similarly, each virtual network request is denoted by a weighted undirected graph $G_V = (N_V, L_V)$, where N_V is the set of virtual nodes and L_V is the set of virtual links. Every virtual node $n_V \in N_V$ has a requirement of CPU capacity and every link $l_V \in L_V$ has a requirement of bandwidth. We denote the CPU capacity requirement of virtual node n_V as $CPU(n_V)$ and the bandwidth requirement of virtual link l_V as $BW(l_V)$.

B. *Virtual Network Embedding Problem*

The VNE problem aims to find a proper mapping method $M_{V \to S}$, which can embed one or more VN requests in a SN. Usually, the mapping process can be divided into two steps, node mapping and link mapping. Node mapping assigns each virtual node to a substrate node which can meet the CPU capacity requirement, i.e. $RC(n_S) \ge CPU(n_V)$ .Link mapping embeds every virtual link in a substrate path. Each virtual link must choose the path with enough bandwidth, i.e. $RB(p_S) \ge BW(l_S)$.

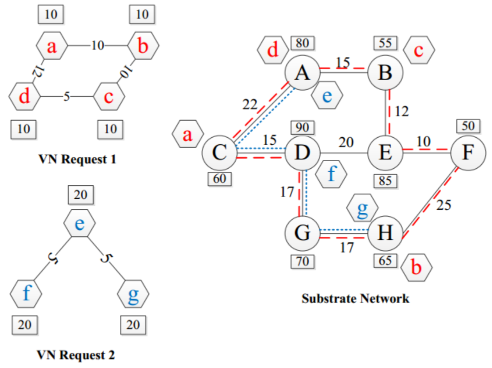

Fig. 1. An example of VNE

Fig. 1 shows an example of virtual network embedding. There are two VN requests in Fig. 1. The node mapping solution of the VN request 1 is $\{a \to C, b \to H, c \to B, d \to$ A}, and the solution for VN request 2 is $\{e \to A, f \to D, g \to H\}$. As for link mapping, the solution for VN request 1 is $\{(a,b) \to (C,D,G,H), (b,c) \to (H,F,E,B), (c,d) \to (B,A), (d,a) \to (A,C)\}$ and the solution for VN request 2 is $\{(e,f) \to (A,C,D), (f,g) \to (D,G,H)\}$.

C. *Objectives*

1) *Increasing Average Revenue*

We use the similar way of previous work [2] to define the revenue of a VN request G_V:

$$Rev(G_V) = \sum_{l_V \in L_V} BW(l_V) + \sum_{n_V \in N_V} CPU(n_V) \qquad (1).$$

So the average revenue of the InP in a long run is defined as:

$$\lim_{T\to\infty}\frac{\sum_{G_V\in SE} Rev(G_V)}{T} \quad (2)$$

where *SE* is the set of VN requests that are successfully embedded.

2) *Increasing the Ratio of Revenue to Cost*

The ratio of revenue to cost in the long run is denoted by *Ratio* (*R*/*C*). Firstly, the cost of allocating a VN to the SN is defined as:

$$Cst(G_V)=\sum_{l_V\in L_V}\sum_{l_S\in L_S} le(l_V,l_S)gBW(l_V)+\sum_{n_V\in N_V} CPU(n_V) \quad (3).$$

Now we can give the formula of *Ratio* (*R*/*C*) in the long run:

$$Ratio(R/C)=\lim_{T\to\infty}\frac{\sum_{G_V\in SE} Rev(G_V)}{\sum_{G_V\in SE} Cst(G_V)} \quad (4).$$

3) *Increasing Request Acceptance Ratio*

The acceptance ratio, denoted by *Ratio*(*AC*), is the ratio between the number of VN requests accepted and the total number of VN requests, i.e.

$$Ratio(AC)=\lim_{T\to\infty}\frac{\sum_{G_V\in SE} |G_V|}{\sum_{G_V\in SR} |G_V|} \quad (5)$$

where *SE* is the set of VN requests that are embedded successfully and *SR* is the set of all the VN requests.

This paper aims to propose an online VNE algorithm. The main objective is to increase the average revenue in the long run. We also want to increase *Ratio* (*R*/*C*) and *Ratio* (*AC*) at the same time.

D. *Constraints*

1) *Capacity Constraints*

Capacity constraints include the node constraint and the link constraint, i.e.

$$\begin{aligned} &ne(n_x,n_i)gCPU(n_x)\le RC(n_i) \quad \forall n_i\in N_S,\forall n_x\in N_V;\\ &le(l_{xy},l_{ij})gBW(l_{xy})\le RB(l_{ij}) \quad \forall l_{ij}\in L_S,\forall l_{xy}\in L_V \end{aligned} \quad (6).$$

2) *Flow-Related Constraints*

When we embed a virtual link l_{xy} in a substrate path p_{ij}, except for the substrate node n_i and n_j, every node on path p_{ij} should have equal amount of flow that enters and leaves it. We can use the following equation to describe such constraint:

$$\forall n_i\in N_S,\forall l_{xy}\in L_V,$$

$$\sum_{l_{ij}\in L_S} le(l_{xy},l_{ij})-\sum_{l_{ji}\in L_S} le(l_{xy},l_{ji})=\begin{cases}1 & if(ne(n_x,n_i)=1)\\ -1 & if(ne(n_y,n_i)=1)\\ 0 & \text{otherwise.}\end{cases} \quad (7).$$

3) *Meta and Binary Constraints*

When we embed a single VN to a SN, every virtual node should find one and only one suitable substrate node, and every substrate node can only hold at most one virtual node i.e.

$$\forall n_x\in N_V,\ \sum_{n_i\in N_S} ne(n_x,n_i)=1;\ \forall n_i\in N_S,\ \sum_{n_x\in N_V} ne(n_x,n_i)\le 1 \quad (8).$$

But in an online VNE problem, a substrate node can hold more than one virtual node as long as these virtual nodes are not in the same VN.

4) *Domain Constrains*

The domain constraints are defined in (9):

$$\begin{aligned} &\forall n_i\in N_S,\forall n_x\in N_V, ne(n_x,n_i)\in\{0,1\};\\ &\forall l_{ij}\in L_S,\forall l_{xy}\in L_V, le(l_{xy},l_{ij})\in\{0,1\} \end{aligned} \quad (9)$$

E. *Problem Formulation*

Section Ⅱ.C gives the three objectives of the online VNE problem. But we need to find a solution for the single VNE problem at first. We aim to reduce the resource cost of VN request in this paper. Since the cost of CPU capacity is fixed, we can only minimize the bandwidth cost of links. Thus the objective function of the VNE-ACS algorithm is

$$\min\sum_{l_{xy}\in L_V}\sum_{l_{ij}\in L_S} le(l_{xy},l_{ij})gBW(l_{xy}) \quad \text{subject to } (6)\text{-}(9) \quad (10)$$

III. PROPOSED VNE-ACS ALGORIYHM

In this section, we first give the solution of node mapping using VNE-ACS and then gives the algorithm of link mapping. At last, we give the overall algorithm for the VNE-ACS.

A. *ACS for Node Mapping*

An ant is a solution of node mapping, and the *k*th ant is denoted by $A_k=\{a_k^1,a_k^2,...,a_k^V\}$. The value of a_k^v is the order number of a substrate node, and the *v*th virtual node is mapped to node a_k^v. We usually embed virtual nodes with more resource requirement earlier in case of resource shortage. *NR* value can represent the resource amount of a node, and for a virtual node n_x it is defined as:

$$NR(n_x)=CPU(n_x)g\sum_{l_V\in L(n_x)} BW(l_V) \quad (11)$$

where $L(n_x)$ is the set of all the links which are adjacent to node n_x. *NR* value for a substrate node n_i is defined as:

$$NR(n_i)=RC(n_i)g\sum_{l_S\in L(n_i)} RB(l_S) \quad (12).$$

In (12), $L(n_i)$ is the set of all the links adjacent to node n_i.

There are two rules to update pheromone, the global rule in (13) and the local rule in (15). At the end of each iteration,

$$\tau_j(t+1)=(1-\rho)g\tau_j(t)+\rho g\Delta\tau_j \quad (13)$$

where $\Delta\tau_j$ is the additional pheromone amount. The value of $\Delta\tau_j$ is calculated by (14)

$$\Delta\tau_j=\begin{cases}Q/f(S_{best}) & \text{if } n_j\in S_{best}\\ 0 & \text{otherwise}\end{cases} \quad (14)$$

where $Q=100gRev(G_V)$ and $f(S_{best})$ is the fitness value of S_{best}.

Once an ant chooses n_j, the pheromone $\tau_j(t)$ will be updated as:

$$\tau_j(t) = (1-\rho')g\tau_j(t) + \rho' g\tau_0 \tag{15}$$

ρ' is the local evaporation parameter.

In VNE-ACS the heuristic information contains both resource message and link distance message. So we define the heuristic information as:

$$\eta_j = \begin{cases} NR(n_j) / \sum_{i \in Ant_k} dis_{ij} & \text{if } RC(n_j) \geq CPU(n_v) \\ 0 & \text{otherwise} \end{cases} \quad (16).$$

In (17), n_v is the virtual node being mapped now and $\sum_{i \in Ant_k} dis_{ij}$ is the sum of shortest distance between node n_j and every substrate nodes which are already in Ant_k. Specifically, $\sum_{i \in Ant_k} dis_{ij}$ is 0 for the first node of any ant. Notice that we set the initial distance of every two different node as 1 if they are adjacent directly, otherwise infinity.

Now we can give the state transition rule. Ant_k chooses a node n_j for mapping the next virtual node n_v by applying:

$$j = \begin{cases} \arg\max_{u \in S} \{(\tau_u)^\alpha g(\eta_u)^\beta\} & \text{if } \quad q \leq q_0 \\ J & \text{otherwise} \end{cases} \tag{17}$$

where S is the set of substrate nodes that can satisfy the resource requirement of n_v. When $q > q_0$, the possibility of choosing n_j is defined as:

$$p_j = (\tau_j)^\alpha g(\eta_j)^\beta / \sum_{u \in S} (\tau_u)^\alpha g(\eta_u)^\beta \tag{18}$$

Algorithm 1 Link Mapping and Feasibility Checking

```
 1:Remove a link $l_{xy}$ from $L_V$ then $L_V = L_V - l_{xy}$
 2:Find the two nodes $n_i$ and $n_j$ which $n_x$ and $n_y$ are embedded in
 3:Remove the substrate links without enough bandwidth for $l_{xy}$ from $G_S$
 4:Find a shortest path between $n_i$ and $n_j$ as the link mapping solution for $l_{xy}$
 5:if no path can be found then
 6:    return UNFEASIBLE
 7:end if
 8:if $L_V$ is empty then
 9:    return FEASIBLE
10:else
11:    go to Step3
12:end if
```

Algorithm 2 Reconstruct a New Solution for Node Mapping

```
1: $A_k = \varnothing$
2:for $v \in (1, V)$ do
3:    Using roulette method to choose a node $n_j$ for $n_v$ to embed, the
      possibility of choosing $n_j$ is $NR(n_j) / \sum_{n_u \in S} NR(n_u)$ where $S$ is the set
      of suitable substrate nodes for $n_v$
4:    $A_k = A_k$ U $\{n_j\}$
5:end for
6:Output $A_k$.
```

B. *Link Mapping and Feasibility Check Method for an Ant*

An ant can only represent the node mapping result, and we cannot promise it is a feasible solution. Similar to [2], we use Algorithm 1 for link mapping and feasibility checking.

Once we find that a solution is unfeasible, we don't discard it at once. We only use a simple way to construct a new one and this can increase the diversity of the solutions set. Algorithm 2 gives the method to reconstruct a new ant.

C. *VNE-ACS Description*

Algorithm 3 shows the overall procedure of the proposed VNE-ACS algorithm. TABLE Ⅱ gives the descriptions of some notations we use in Algorithm 3.

TABLE II. NOTATIONS USED IN ALGORITHM 3

Notation	Description
S_{best}	The global best solution so far
BestFitness	The fitness value of S_{best}
ITE	The iteration times of the algorithm
ANT	The number of ants
A_k	The *k*th ant
V	The number of VN nodes
C	The maximum counts of reconstructing an ant
$f(A_k)$	The fitness value of A_k

Algorithm 3 VNE-ACS Algorithm

```
 1: $S_{best} = \varnothing,\ BestFitness = +\infty$
 2:for $t \in [1, ITE]$ do
 3:    for $k \in [1, ANT]$ do
 4:        $A_k = \varnothing$ , FAIL=0
 5:        for $v \in [1, V]$ do
 6:            generate a random $q \in (0,1)$
 7:            if $q>q_0$ then
 8:                compute $p_j$ for every $n_j \in S$ using (18)
 9:            end if
10:            choose a substrate node $n_j$ for $n_v$ to embed in using (17)
11:            if there is no suitable $n_j$ then
12:                FAIL=1
13:                go to Step 22
14:            end if
15:            $A_k = A_k \cup \{n_j\}$
16:            apply local pheromone update rule in (15) on $n_j$
17:        end for
18:        check the feasibility of $A_k$ using Algorithm 1
19:        if $A_k$ is UNFEASIBLE then
20:            FAIL =1
21:        end if
22:        if FAIL =1 then
23:            for $c \in [1, C]$ do
24:                Reconstruct $A_k$ using Algorithm 2
25:                if $A_k$ is FEASIBLE then
26:                    FAIL =0
27:                    go to Step 31
28:                end if
29:            end for
30:        end if
31:        if $f(A_k) <$ BestFitness then
32:            $S_{best} = A_k$
33:            BestFitness=$f(A_k)$
34:        end if
35:    end for
36:    apply global pheromone update rule in (13) on all substrate nodes
37:end for
38:If $BestFitness \neq +\infty$ , $S_{best}$ is the optimal solution of node mapping and
   we use Algorithm 1 to obtain the link mapping result. Otherwise, the
   VN request is rejected.
```

IV. Performance Evaluation

In this section, we evaluate two kinds of VNE problems, one is for single VN request and the other is for online VN requests. Section Ⅳ.A describes the simulation settings. For single VN request, Section Ⅳ.B compare VNE-ACS with two other algorithms [2, 13] in terms of the cost of VN request. And Section Ⅳ.C shows that, in the online VNE problem, our algorithm is better than the other two algorithms [2, 13] using the metrics we proposed in Section Ⅱ.C.

A. Simulation Settings

All the network topologies used in this paper are generated by the Georgia Tech Internet Topology Modeling (GT-ITM) tool [2]. There are two kinds of substrate network we use in this paper, one 100 nodes and the other has 200 nodes. The connectivity of both of the two substrate networks is 0.1. The CPU capacity of a substrate node is an integer uniformly distributed between 50 and 100. And the bandwidth for a substrate link is also a uniform distribution integer. When the resource is sufficient, the bandwidth is between 50 and 100. But when the resource is not so adequate, the value is between 0 and 100.

In the online VNE problem, for each VN request, the number of virtual nodes is an integer between 2 and 20 and the connectivity is 0.5. Both the CPU requirement of virtual nodes and the bandwidth of virtual links are integers between 0 and 50. All the random variables mentioned in VN request follow uniform distribution.

For our algorithm VNE-ACS, we perform many times of experience to find out the best assignments for its parameters and other settings and the results are listed in TABLE Ⅲ.

TABLE III. Some Assignments about VNE-ACS

Notation	Description	Assignment
α	The importance parameter of pheromone	1
β	The importance parameter of heuristic information	0.9
ρ	The global pheromone evaporation parameter	0.1
ρ'	The local pheromone evaporation parameter	0.1
q_0	The probability using in (17)	0.9
τ_0	The initial pheromone value	100
K	The number of ants	5
ITE	The iteration times	20
C	The maximum counts of reconstructing an ant	50

We choose two algorithms to make comparisons for both single VN request and online VN requests. The first algorithm [2] is a PSO-based algorithm which is called VNE-UEPSO. VNE-UEPSO only considers the *NR* value for node mapping. The other algorithm [13] is an ACO-based algorithm called E-ACO-SNP which also only considers the resource of a node in node mapping. And E-ACO-SNP uses the model of ACO but not ACS.

B. Evaluation for Single VN Request

We choose 6 VN requests randomly for our simulation as 6 test cases, and the substrate network we use has 100 nodes.

TABLE Ⅳ shows the comparison result of the three algorithms. We run each algorithm 30 times for every test case and record results for these 30 experiments. Then we give the average number (Ave), the standard deviation (Std) and the minimum value (Best) of the 30 results. We also give the tow-tailed test for these results at the 0.05 level. That is to say if the absolute value of *t-test* is larger than a certain value, then we can be at least 95% sure that the difference between the two groups of results we tested is significantly different.

TABLE IV. The Comparison Result of Single VN Request

Test Case	Algorithm	Ave	Std	Best	T-test
1	VNE-ACS	862.60	32.83	818	
	VNE-UEPSO	1105.07	40.85	1030	**-22.07***
	E-ACO-SNP	1125.93	39.65	1029	**-31.12***
2	VNE-ACS	868.07	27.28	818	
	VNE-UEPSO	1145.07	42.29	1058	**-31.51***
	E-ACO-SNP	1148.00	43.95	1065	**-31.62***
3	VNE-ACS	5840.53	218.52	5260	
	VNE-UEPSO	6048.93	144.58	5680	**-3.73***
	E-ACO-SNP	6102.17	135.24	5804	**-5.85***
4	VNE-ACS	5596.03	201.93	5157	
	VNE-UEPSO	5791.47	133.02	5456	**-3.88***
	E-ACO-SNP	5808.47	159.28	5405	**-4.62***
5	VNE-ACS	6670.07	167.71	6296	
	VNE-UEPSO	6667.43	152.83	6363	0.06
	E-ACO-SNP	6723.80	122.22	6419	-1.38
6	VNE-ACS	5895.03	184.28	5539	
	VNE-UEPSO	6172.57	127.32	5943	**-6.74***
	E-ACO-SNP	6217.63	133.93	5838	**-7.41***

From TABLE Ⅳ, we can see that except for test case 5, the average and minimum fitness values of VNE-ACS are better than the other two algorithms'. Even though the results of VNE-UEPSO are better than the results of VNE-ACS in test case 5, there is no significant difference between them according to the *t-test* value. Every *t-test* value tailed with asterisk (*) represents that the difference of the results of this algorithm is significant different from the results of VNE-ACS at the confidence level of 95%. And the absolute value of t-test value is larger, the confidence level is higher.

To sum up, in the single VN request problem, the performance of VNE-ACS is significantly better than the two compared algorithms in most cases.

C. Evaluation for Online VN Requests

We first give the simulation description of online VN requests. There are 50,000 time units in total. The arrivals of VN requests follow a Poisson distribution, and there are 5 VN requests per 100 time units on average. We model the lifetime of a VN using an exponential distribution with a mean of 500 time units. So it can be about 2,500 VN requests in the whole simulation. We simulate online VN requests on several kinds of SNs mentioned in Section Ⅳ.A. The results of these different simulations are similar, so we only show one of them in this paper. The SN we used has 100 nodes and the bandwidth of every link varies between 50 and 100.

Fig. 2, Fig. 3 and Fig. 4 show the comparison results of average revenue in the long run, the ratio of revenue to cost in the long run, and the request acceptance ratio respectively.We can see from these figures that the values of three metrics nearly come to stable states after 10,000 time units. So we evaluate these three algorithms according to the data after 10,000 time units. It is obvious that VNE-ACS is better than

VNE-UEPSO and E-ACO-SNP in terms of three kinds of metrics mentioned in Section Ⅱ.C.

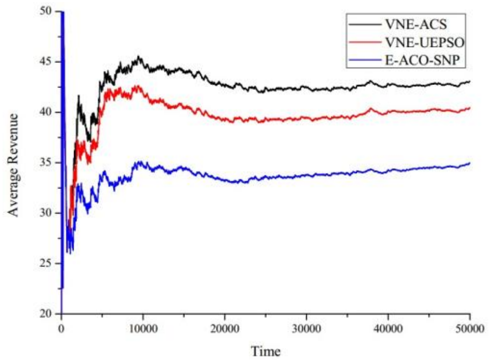

Fig. 2. The Comparison Results for Average Revenue

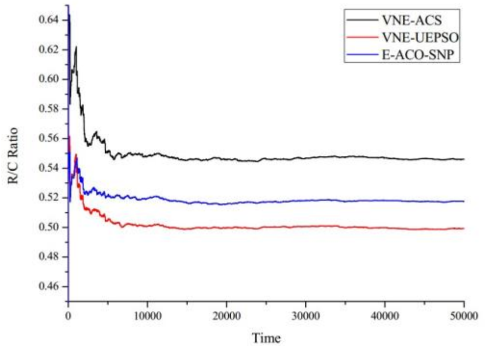

Fig. 3. The Comparison Results for R/C Ratio

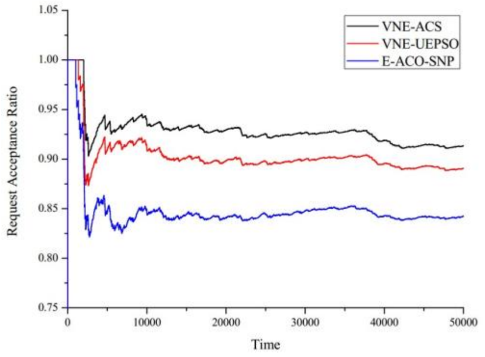

Fig. 4. The Comparison Results for Request Acceptance Ratio

V. CONCLUTION

The virtual network embedding problem is a core problem in network virtualization. It is an NP hard problem and has many constraints as well. Traditional algorithms cannot solve this problem when there are thousands of VN requests or more. In this paper, we proposed an ant colony system based algorithm VNE-ACS in order to increase average revenue of the online VNE problem in the long run. Different from the previous work, we take link distance message into consideration in node mapping phase. We compared our algorithm with VNE-UEPSO and E-ACO-SNP for both single VN request and online VN requests. The experimental results show that VNE-ACS outperforms the other two algorithms in some aspects.

There are still many challenges in the VNE problem. Green computing is more and more important these days, but most work related to VNE aims to maximize the revenue or accept ratio without considering the energy cost. Security is also a factor we usually ignore in the VNE problem. So in our future work, we will define some new metrics and emphasize on objectives such as energy-saving and security enhancing.

REFERENCES

[1] Chowdhury, N. M. M. K., and R. Boutaba. "Network virtualization: state of the art and research challenges." IEEE Communications Magazine47.7(2009):20-26.

[2] Zhang, Zhongbao, et al. "A unified enhanced particle swarm optimization-based virtual network embedding algorithm." International Journal of Communication Systems 26.8(2013):1054–1073.

[3] Zhu, Y., and M. Ammar. "Algorithms for Assigning Substrate Network Resources to Virtual Network Components." Proceedings - IEEE INFOCOM (2006):1-12

[4] Fan, J., and M. H. Ammar. "Dynamic Topology Configuration in Service Overlay Networks: A Study of Reconfiguration Policies." Proceedings - IEEE INFOCOM 2.9(2006):1-12.

[5] Houidi, I., W. Louati, and D. Zeghlache. "A Distributed Virtual Network Mapping Algorithm." IEEE International Conference on CommunicationsIEEE, 2008:5634-5640.

[6] Chowdhury, N. M. M. K., M. R. Rahman, and R. Boutaba. "Virtual Network Embedding with Coordinated Node and Link Mapping."Proceedings - IEEE INFOCOM 20.1(2009):783-791.

[7] Davis, Lawrence. "Handbook of genetic algorithms." Handbook of Genetic Algorithms (1991).

[8] Kirkpatrick, Scott. "Optimization by Simulated Annealing: Quantitative Studies." Journal of Statistical Physics 34.5(2010):975-986.

[9] Kennedy, James, and R. Eberhart. "Particle swarm optimization." (1995).

[10] Colorni, Alberto, M. Dorigo, and V. Maniezzo. "Distributed Optimization by Ant Colonies." European Conference on Artificial Life 1991.

[11] X.-M. Hu, J. Zhang, H. S.-H. Chung, Y. Li, and O. Liu, "SamACO: variable sampling ant colony optimization algorithm for continuous optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* vol. 40, no. 6, pp. 1555-1566, 2010.

[12] Q. Yang, W.-N. Chen, Z. Yu, T. Gu, Y. Li, H. Zhang, and J. Zhang, "Adaptive multimodal continuous ant colony optimization," *IEEE Transactions on Evolutionary Computation,* vol. 21, no. 2, pp. 191 - 205, 2017.

[13] Chang, Xiaolin, et al. "Green cloud virtual network provisioning based ant colony optimization." Conference Companion on Genetic and Evolutionary Computation 2013:1553-1560.

[14] Fajjari, Ilhem, et al. "VNE-AC: Virtual Network Embedding Algorithm Based on Ant Colony Metaheuristic." Communications (ICC), 2011 IEEE International Conference on IEEE, 2011:1-6.

[15] W.-N. Chen, and J. Zhang, "An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 39, no. 1, pp. 29-43, 2009.

[16] W.-N. Chen, and J. Zhang, "Ant colony optimization for software project scheduling and staffing with an event-based scheduler," *IEEE Transactions on Software Engineering,* vol. 39, no. 1, pp. 1-17, 2013.

[17] W.-N. Chen, J. Zhang, H. S.-H. Chung, R.-Z. Huang, and O. Liu, "Optimizing discounted cash flows in project scheduling—an ant colony optimization approach," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 40, no. 1, pp. 64-77, 2010.

[18] Dorigo, M., and L. M. Gambardella. "Ant colony system: a cooperative learning approach to the traveling salesman problem." IEEE Transactions on Evolutionary Computation 1.1(1997):53-66.