Available online www.jocpr.com

Journal of Chemical and Pharmaceutical Research, 2014, 6(7):327-337

Research Article

ISSN : 0975-7384
CODEN(USA) : JCPRC5

A modified ant colony optimization algorithm for virtual network embedding

Fangjin Zhu* and Hua Wang

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

---

ABSTRACT

Traditional Internet architecture is far too rigid for use with large numbers of network applications with different quality of service requirements. One new and promising approach to overcome the rigidity is network virtualization (NV), which allows multiple heterogeneous virtual networks to coexist on a shared substrate network (SN). However, one of the key problems for NV is the virtual network embedding (VNE) problem, which concerns the efficient mapping of virtual nodes and links to SN nodes and paths. The VNE problem has proven to be nondeterministic polynomial-time hard and approximation algorithms are needed to address it. In this paper, we define the VNE problem based on the multiple-choice knapsack model and propose a modified ant colony optimization algorithm to solve the problem. The combination of revenue and acceptance ratio of an SN is used as an important component when designing the fitness function to evaluate iterative solutions obtained by ants, and pheromone update rules are designed based on the fitness function. The cost of a candidate network is defined as the selection heuristic information. Simulation results show that this algorithm performs well with various numbers of VN requests. The algorithm also provides better optimization performance than existing algorithms.

Key words: Network virtualization; virtual network embedding; multi-choice knapsack problem; ant colony optimization; greedy algorithm

---

INTRODUCTION

Numerous network applications that provide different levels of quality of service over the Internet have been developed in the past decades. These applications have various resource requirements and resource usage methods. Traditional Internet architecture is too rigid to adequately support these applications. Network virtualization (NV) [1] is a promising scheme that separates the roles of Internet infrastructure providers (InPs) and service providers (SPs). SPs are then allowed to construct their own virtual networks (VNs) by leasing resources from substrate networks (SNs) of the InPs. Due to their different resource requirements and different operation protocols, these VNs are heterogeneous. Thus, multiple heterogeneous VNs coexist on a shared SN, and the InP needs to allocate VN nodes and links on its SN efficiently to increase revenue. This is referred to as the virtual network embedding (VNE) problem. The VNE problem has proven to be nondeterministic polynomial-time hard (NP-hard) [2]. The NVE problem must be addressed before NV can be viably deployed over the Internet.

There have been many attempts to address the VNE problem [1-11]. Most methods, such as [10] and [11], divide the VNE problem into two stages: first solve node mapping using greedy algorithms, and then search for physical paths for link mapping. These methods ignore the relationship between node mapping and link mapping in order to reduce the complexity of the problem, but provide only poor performance due to the restriction of the solution space.

To solve node mapping and link mapping simultaneously and get better performance, Chowdhury et al. [2] proposed ViNEYard—virtual network embedding algorithms with coordinated node and link mapping. In their algorithms, a mixed integer programming model is formulated, and a corresponding relaxed linear programming problem is solved for node mapping (deterministic or randomized), and another iteration of the linear programming problem is solved for link mapping. Simulation results showed that ViNEYard achieved better performance than existing solutions to the VNE problem.

The above algorithms focus on online VN requests, trying to minimize the cost of each VN request to make an SN accept more VN requests, and thus maximize the revenue of the InP accordingly. These online algorithms are lack of knowledge of successive requests, and greedy ideas are used to address the current request, and result in poor performance in some situations. Therefore, mapping migration [11] and time window algorithms [2] have been proposed. Using a batch process and a simple selection strategy, better performance can be obtained, but there is much room for improving the optimization performance.

This paper focuses on processing multiple VN requests simultaneously. Based on the multiple-choice knapsack problem (MCKP) model, we propose a modified ant colony optimization (ACO) algorithm. In our algorithm, a set of candidate physical networks are first found for each VN request, and the VNE problem is converted to an MCKP problem. The optimization goal is to maximize revenue and achieve optimal load-balancing characteristics, so that the SN can support more VN requests. We designed a fitness function to evaluate the solutions obtained by ants in each iteration. To hasten convergence, the cost of a physical network is used as the heuristic information for selecting one candidate physical network. According to the characteristics of MCKP model, two types of pheromone are defined and the corresponding pheromone update rules are proposed. Simulation results show that our proposed algorithm has better optimization performance than the greedy algorithms in literature [2].

This paper is organized as follows: Section 2 describes the background of the VNE problem and defines the optimization problem to be addressed; Section 3 proposes the modified ACO algorithm to solve the optimization problem; Section 4 provides simulation results and analysis; and Section 5 provides a conclusion.

BACKGROUND AND DEFINITIONS

In this section we first describe the VNE problem by an example, then model the SN and VN request as undirected graphs, finally give a mathematical model based on some definitions.

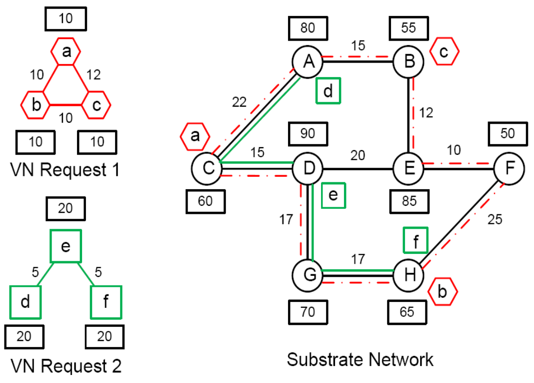

Fig. 1: Illustration of virtual network embedding [2, 11]

Fig. 1 [2, 11] illustrates one SN (right) of an InP, and two VN requests (left) to be embedded in it. For a VN request, each VN node is mapped to an SN node that satisfies the resource constraints of the VN node, and each VN link is mapped to an SN loop-free path that also satisfies the bandwidth constraints of the VN link. For VN request 1, the VN nodes (a, b, c) are mapped to the SN nodes (C, H, B), respectively, and VN links ((a, b), (a, c), (b, c)) are mapped to the SN paths ((C, D, G, H), (C, A, B), (H, F, E, B)). Because all SN nodes and paths can satisfy the resource requirements of all VN nodes and links, VN request 1 can be accepted. For the same reason, VN request 2 is also accepted. These accepted VN requests can bring revenue for the InP (the owner of the SN) as they consume SN resources. If no SN nodes or paths satisfy the resource requirements, the VN request will be rejected. When many VN requests are made, different mapping methods accept and reject different VN requests, which influences the usage of the SN and results in different amounts of revenue for the InP. This is the essence of the VNE problem. The optimization goal of the VNE problem is to sufficiently use the SN resources by selecting an appropriate mapping (or embedding) for each VN request so as to maximize the entire revenue of the InP.

The SN is modeled as an undirected weighted graph $G^S = (V^S, E^S)$, where V^S is the set of SN nodes and E^S is the set of SN links. Each node $v \in V^S$ has an attribute set $C_{attr}^{S,V}(v)$ to describe its capacity. In this paper, $C_{cpu}^{S,V}(v)$ is used to describe the CPU capacity of node v, which is in a rectangle near it, and $C_{loc}^{S,V}(v)$ is used for the location information. Each edge $e \in E^S$ also has an attribute set $C_{attr}^{S,E}(e)$; in this paper, $C_{bw}^{S,E}(e)$ is used to describe the bandwidth of the link, which is a real value near the link. The VN request is also modeled as an undirected weighted graph $G^R = (V^R, E^R)$, and the upper character S is replaced with R for each attribute donation.

For a VN request R_i, the revenue of the InP is fixed if it is accepted. The revenue can be defined as follows:

$$\mathrm{Re}(R_i) = \alpha \sum_{v \in V^{R_i}} C_{cpu}^{R_i,V}(v) + \beta \sum_{e \in E^{R_i}} C_{bw}^{R_i,E}(e) \qquad (1)$$

where α, β reflect the relative importance of CPU capacity and link bandwidth, respectively.

The cost of a VN request R_i will differ for different mappings. Assume that there are K mappings $M_{i,j}(j = 1,2,...,K)$ for R_i. $P(e)$ is the SN path for VN link e, and $\|P(e)\|$ is the length of $P(e)$. For each mapping $M_{i,j}(j = 1,2,...,K)$, the cost of R_i is defined as:

$$C_{M_{i,j}}(R_i) = \alpha \sum_{v \in V^{R_i}} C_{cpu}^{R_i,V}(v) + \beta \sum_{e \in E^{R_i}} C_{bw}^{R_i,E}(e) \|P(e)\| \qquad (2)$$

where α, β have the same meaning with (1).

For a VN request R_i, let the K mappings $M_{i,j}(j = 1,2,...,K)$ be the candidate physical networks (or, for brevity, “candidate networks”) for it.

Given N_R VN requests, due to resource constraints, only some requests can be accepted. The optimization goal is to maximize the revenues of the InP. If a VN request R_i is accepted, only one candidate network is used. A binary variable $x_i^j = 1(j = 1,2,...,K)$ denotes that the j^{th} candidate network is selected for R_i, and $\sum_{j=1}^{K} x_i^j = 1$ guarantees that only the j^{th} candidate network is selected. If this VN R_i is rejected, then $\sum_{j=1}^{K} x_i^j = 0$.

The j^{th} candidate network $M_{i,j}$ of a VN request R_i is a sub-graph of SN, and it contains information of node mapping and link mapping. For an SN node $v \in V^S$, if it is used by $M_{i,j}$, then a binary variable $M_{i,j}(v) = 1$. Assumed its corresponding virtual node is $v' \in V^{R_i}$, then $C_{cpu}^{M_{i,j},V}(v) = C_{cpu}^{R_i,V}(v')$. For an SN edge $e \in E^S$, if it is used by $M_{i,j}$, then a binary variable $M_{i,j}(e) = 1$. Assumed its corresponding virtual edge is $e' \in E^{R_i}$, then $C_{bw}^{M_{i,j},E}(e) = C_{bw}^{R_i,E}(e')$. Therefore, the optimization problem is defined by the following mathematical model:

$$Max \ \sum_{i=1}^{N_R} (\mathrm{Re}(R_i) \times \sum_{j=0}^{K} x_i^j) \qquad (3)$$

$s.t.$

$$\sum_{i=1}^{N_R} \sum_{j=1}^{K} x_i^j M_{i,j}(v) C_{cpu}^{M_{i,j},V}(v) \le C_{cpu}^{S,V}(v), \ \forall v \in V^S \qquad (4)$$

$$M_{i,j}(v)=\begin{cases}1, & \text{if } v\in V^S \text{ is used by } M_{i,j}\\ 0, & \text{else}\end{cases} \quad (5)$$

$$C_{cpu}^{M_{i,j},V}(v)=C_{cpu}^{R_i,V}(v'),\ M_{i,j}: v'\in V^{R_i}\to v\in V^S \quad (6)$$

$$\sum_{i=1}^{N_R}\sum_{j=1}^{K}x_i^j M_{i,j}(e)C_{bw}^{M_{i,j},E}(e)\le C_{bw}^{S,E}(e),\ \forall e\in E^S \quad (7)$$

$$M_{i,j}(e)=\begin{cases}1, & \text{if } e\in E^S \text{ is used by } M_{i,j}\\ 0, & \text{else}\end{cases} \quad (8)$$

$$C_{bw}^{M_{i,j},E}(e)=C_{bw}^{R_i,E}(e'),\ M_{i,j}: e'\in E^{R_i}\to e\in E^S \quad (9)$$

$$\sum_{j=1}^{K}x_i^j\le 1,\ \ \forall i=1,2,...,N_R \quad (10)$$

$$x_i^j\in\{0,1\},\ \forall i=1,2,...,N_R,\ \forall j=1,2,...,K \quad (11)$$

where (3) is the optimization goal, (4), (5) and (6) together describe the node resource constraints of the SN, (7), (8) and (9) together describe the edge resource constraints of the SN, and (10) and (11) reflect whether a VN request is accepted or rejected in the VNE problem.

The above mathematical model indicates that each accepted request has K choices (candidate networks) and only one choice is selected; furthermore, only some requests can be accepted due to resource constraints. Which requests are accepted and which candidate network is selected for an accepted request is the essence of the VNE problem. So a request has $K+1$ choices (rejected, or one of K candidate networks is selected), and there are $(K+1)^{N_R}$ solutions for our VNE problem. The time complexity of enumerating all solutions and selecting the best one is exponential and it is unlikely to address it when the number of request N_R becomes larger. This problem is a multiple choice knapsack problem and has proven to be NP-hard [12]. Approximation algorithms are needed to tackle this optimization problem. The research works [2-11] adapted various greedy algorithms, however, the optimization performance are poor in some cases through only one iteration process. In our paper, more random iteration processes are adapted and the knowledge gained by one iteration process can be used in following iterations so that better optimization performance can be obtained.

To evaluate a solution s of one random iteration process and guide the following iteration processes, a fitness function is needed. A fitness function represents our greedy ideas to get better results. According to the optimization goal of the VNE problem, a solution with more revenue and higher acceptance rate is preferred. So we designed a fitness function that considers both the revenue and the number of accepted VN requests:

$$f(s)=\phi\sum_{i=1}^{N_R}(\mathrm{Re}(R_i)\times\sum_{j=0}^{K}x_i^j)+\varphi\sum_{i=0}^{N_R}\sum_{j=0}^{K}x_i^j \quad (12)$$

where ϕ,φ reflects the relative importance of revenue and the acceptance ratio. Simulation results show that our fitness function also introduces better network load balance.

A MODIFIED ACO ALGORITHM

Based on the above mathematical model of the VNE optimization problem and our analysis, an algorithm with more random iteration processes is needed. The ACO algorithm, first proposed by Dorigo [13], is a meta-heuristic algorithm and has been successfully used in many combinational optimization problems. In our paper, the process of searching for candidate networks is first given, and then a modified ACO algorithm is proposed to tackle the VNE problem based on the MCKP model.

1. Process of searching for candidate networks for a VN request

In our algorithm K candidate networks are pre-computed for each VN request. Many researchers have proposed ways to embed a VN request [1, 2, 10, 11]. We can use these algorithms to construct K candidate networks separately for each request. In our paper, we use the idea from the ViNEYard solution of Chowdhury et al. [2] to search for candidate networks for a VN request. In the ViNEYard solution, multiple paths are selected for a pair of nodes and the data is split to transfer through these paths. This couples the VNE solution with the bottom data transmission protocols strongly and brings some other problems such as out of sequence. To overcome these shortcomings, we use only one path for a pair of nodes, and use their intermediate results to construct K candidate networks.

First, a linear programming model is solved, for which the optimization goal is relevant to both minimizing the cost of a VN request and providing load balancing for the SN. Each SN node is given a probabilistic selection value for a VN node, and multiple SN paths with their probabilities are obtained for each pair of the corresponding SN nodes.

Second, an SN node is randomly selected for each VN node, and each VN link is checked to see if an SN path satisfies its bandwidth constraints. If all VN links are mapped to SN paths successfully, a candidate network is found. The process is repeated until K candidate networks are found for a VN request.

2. Solution construction process of an ant
Each ant constructs an entire solution of the VNE problem based on the MCKP model individually. It first selects a VN request according to the probability defined in (15) and then selects a candidate network of this VN request according to the probability defined in (16). If the candidate network can be embedded in the SN, i.e., the residual CPU capacities of the SN nodes and the residual bandwidth resources of the SN edges can satisfy with the resource requirements of the candidate network, the VN request is accepted and the candidate network is used as its mapping; otherwise the VN request is rejected. If a VN request is accepted, the corresponding resources of its selected candidate network are reduced in the SN. The process is repeated until all requests are processed; a solution is then obtained by the ant. After each ant finishes its solution construction process, updates the pheromone trail, and starts the next iteration, until the terminal condition of the algorithm is reached. The probabilities of selection are determined by pheromone and heuristic information. Pheromone is defined as follows:

Pheromone: Based on the idea of MCKP, two types of pheromones are defined: $\rho_i (i=1,2,...,N_R)$ is defined as the expectation that a VN request $R_i (i=1,2,...,N_R)$ will be accepted, and $\rho_{ij} (i=1,2,...,N_R, j=1,2,...,K)$ is defined as the expectation that the j^{th} candidate network of R_i is selected for the mapping of R_i.

Based on the characteristics of MCKP, at most one candidate network will be selected for a VN request, which indicates that when one candidate network of a VN request has higher probability to be selected (has higher pheromone value), this VN request also has higher probability to be accepted (also has higher pheromone value); therefore, the relation between these two types of pheromone can be defined as follows:

$$\rho_i = \max\left(\rho_{ij}\right),\ i=1,2,...,N_R,\ \ j=1,2,...,K \quad (13)$$

Heuristic information for selecting a candidate network: Because the entire optimization goal is to accept as many VN requests as possible, a candidate network with less cost is preferred. Thus, the heuristic information for selecting a candidate network is defined as the inverse of its cost to overcome the initial blindness of the ACO algorithm, which indicates that the candidate network with less cost should be selected with higher probability:

$$\tau_{ij} = 1/C_{M_{i,j}}(R_i),\ i{=}1,2,...,N_R,\ j{=}1,2,...K \qquad (14)$$

Probabilities of two selection operations: The solution construction process includes two selection operations. The first operation selects a VN request and the second selects a candidate network for the VN request. The probability that a VN request R_i will be selected is defined as follows:

$$p_i = \frac{\rho_i}{\sum_{i=1}^{N_R} \rho_i}, \quad i=1,2,...,N_R \qquad (15)$$

As defined in (13), a request R_i which has higher pheromone value ρ_i indicates that at least one of its candidate networks resulted in better performance in last iteration processes, and is selected with higher probability in current iteration process.

The probability that the j^{th} candidate network will be selected for R_i is defined as follows:

$$p_{ij} = \frac{\rho_{ij}{}^{\psi}\tau_{ij}{}^{\zeta}}{\sum_{j=1}^{K}\rho_{ij}{}^{\psi}\tau_{ij}{}^{\zeta}},\ i = 1, 2, ..., N_R,\ \ j = 1, 2, ..., K \quad (16)$$

where ψ, ζ reflect the relative importance of pheromone and heuristic information, respectively, in the selection of the candidate network. A candidate network has higher pheromone value when it was selected in the last iteration processes and better results have been obtained, so it will be selected in current iteration process with higher probability. To accept more requests, a candidate network with less cost is preferred, and this is reflected in the definition of heuristic information (14). So a candidate network with higher pheromone value and higher heuristic information value will be selected with higher probability as (16).

3. Pheromone update rules
After all ants finish their solution construction processes, the fitness function (12) is used to evaluate their solutions, and the best one—the one with the largest fitness value—is selected to update the candidate network selection pheromone. The pheromone is increased for all candidate networks embedded in the SN. The pheromone update rule for each candidate network is defined as follows:

$$\rho_{ij} = (1-\lambda)\rho_{ij} + f(s)\times x_i^j, \\ i = 1, 2, ..., N_R,\ \ j = 1, 2, ..., K \quad (17)$$

where $0 < \lambda < 1$ is the evaporation ratio of the pheromone. The first part of (17) indicates that the past knowledge will decrease their influence on the future iteration processes in an exponential manner, and the value of λ decides their decrement ratio. The second part of (17) indicates that at most one candidate network of a request will get increment of pheromone value. According to an ant's solution construction process, when a request is accepted, one of its candidate networks is selected and the selected candidate network will get pheromone increment if it is in the best solution. Otherwise, none of its candidate networks will get pheromone increment.

After the pheromones of all candidate networks are updated, the pheromones of all VN requests are updated according to (13). Through updating the pheromone values, the knowledge of last iteration processes are accumulated to guide the next iteration process to get better results.

4. Terminal conditions
The ACO algorithm is an evolutionary algorithm and requires multiple iterations. Given a suitable window size, the most optimal (maximal) values obtained in each iteration process are recorded in this window. The variance of these values is used to define the terminal condition. When variance is less than a given threshold, the algorithm is assumed to have reached convergence.

Based on the above definitions, the ACO algorithm, VM-ACO (VNE based on MCKP), which is used to address the optimization problem, is given below:

Step 1. Initialize parameters and search for K candidate networks for each VN request.
Step 2. If the terminal condition is reached, go to step 7; otherwise, start the next iteration.
Step 3. For each ant, select a VN request based on the probability defined in (15) and select a candidate network for the selected VN request according to the probability defined in (16).
Step 4. If the selected candidate network can be embedded in the SN, the VN request is accepted and the candidate network is used as the SN mapping, and the resources are reduced in the SN; otherwise, the selected VN request is rejected.
Step 5. Repeat the selection process until all VN requests are processed, and then obtain a solution.
Step 6. Evaluate all solutions obtained by the ants and select the best solution using (12), update pheromone based

on the rules defined in (17), and proceed to Step 2.
Step 7. Select the best solution as the final VNE solution, output the mapping between the VN request and candidate network, and calculate the acceptance ratio and the entire revenue of SN (InP).

SIMULATION AND ANALYSIS

In this section we describe the simulation environment followed by the main evaluation results. To compare our algorithms and the existing ones fairly, we use the same simulation settings as [2], and select the window-based batch process algorithm WiNE proposed in [2] because it has better optimization performance than other pure online algorithms.

1. Simulation settings

The SN topologies used in our algorithm are generated by the Georgia Tech Internetwork Topology Models (GT-ITM) tool [14] with 50 nodes in a 25 × 25 grid. Each pair of SN nodes is connected randomly with a probability of 0.5. 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 2–10 nodes in the same grid. Each pair of VN nodes is connected randomly with a probability of 0.5, and the CPU and bandwidth resources of VN nodes and VN links are uniformly distributed in the ranges 0–20 and 0–50, respectively.

The VN requests arrive according to a Poisson process and we adjust the average rate from 4 to 8 each 100 time units to change the numbers of VN requests in the network. The lifespan of a VN request is exponentially distributed and the average is 1000 time units. The simulation time is fixed at 50000 time units. We suppose that after 30000 time units, VN requests become stable, and conduct request sampling every 1000 time units. When the simulation time reaches 50000 time units, 20 samples can be obtained.

The algorithm WiNE [2] is a window-based batch process algorithm for the VNE problem, and uses a greedy idea to batch process the requests in descending order of revenue. We use the requests obtained in each sample as the input for WiNE and our algorithm, and compare the average acceptance ratio, the average revenue, the average cost, and the average resource utilization ratio of these two algorithms.

2. Performance metrics

We define six performance metrics to examine the performance of our algorithm and the WiNE algorithm [2]. All these metrics are firstly defined and calculated in each sample, and the average of the 20 samples as the final results to be compared.

The acceptance rate measures the percentage of VN requests accepted by the algorithms.

The generated revenue measures the total revenue of all accepted requests in a sample (the revenue of an accepted request is defined in (1)).

The provision cost measures the provisioning cost of embedding a VN request (defined in (2)).
The node (link) utilization rate measures the CPU (bandwidth) usage percentage of an SN node (link) when the algorithms embed some VN requests in the SN.

To evaluate the characteristics of load balancing of embedding algorithms, we define the *load balancing factor* as the variance of utilization ratios of all nodes and links. This load balancing factor can reflect the fluctuation degree of the resource usage of nodes and links, and lower value of this metric means more balancing that also implies more VN requests can be accepted and more revenue can be obtained.

Because the batch process algorithm WiNE proposed by Chowdhury et al. [2] uses a greedy idea, it is called greedy algorithm in this paper, compared with our VM-ACO algorithm.

3. Optimization performance

Fig. 2 gives the average acceptance rates of the two algorithms when the VN request arrival rate changes from 4 to 8 per 100 time units. Higher VN request arrival rate means more VN requests to be embedded in the SN.

As fig. 2 shows, the acceptance rates of the two algorithms become lower when there are more VN requests. The reason behind is that the nodes and links of the SN have resource constraints and only some of the VN requests can be accepted. However, the numbers of accepted requests become larger with the increment of the arrival rate due to the uniformly requests distribution.

For each VN request arrival rate, the VN requests in each sample are the same and as the input of the two algorithms. The proposed VM-ACO algorithm obtains a higher average acceptance rate than the greedy algorithm of Chowdhury et al. This is because the greedy algorithm only accepts VN requests according to a simple greedy idea: it accepts VN requests starting with those that provide the highest revenue to those that provide the lowest one. However, the VM-ACO algorithm randomly selects VM requests according to pheromone and heuristic information, and then uses an iterative process to optimize performance. Because K candidate networks are preselected, the more VN requests in the network, the higher the acceptance rate of the VM-ACO algorithm.

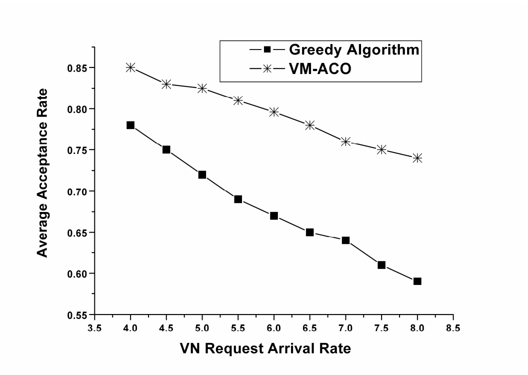

Fig. 2: Average acceptance rate with different VN arrival rates

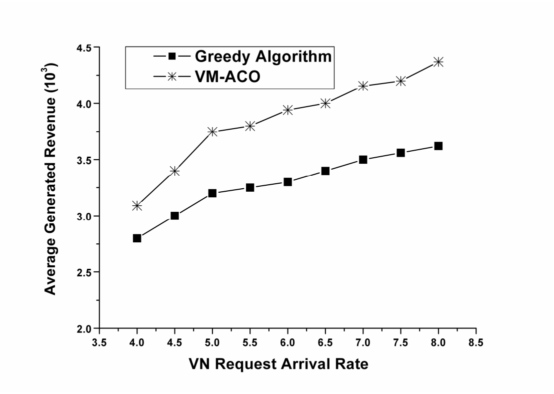

Fig. 3: Average generated revenue with different VN arrival rates

Fig. 3 gives the average generated revenue of the two algorithms when the VN request arrival rate changes from 4 to 8 per 100 time units. For each sample, the generated revenue is the total revenue of all accepted requests, and the average generated revenue is the average of 20 samples.

As fig. 3 shows, the two compared algorithms obtain more average generated revenue with more VN requests. This is because more VN requests give the algorithms more opportunities to embed these requests efficiently. More revenue along with higher acceptance rate imply that the two algorithms can actually embedded requests that generate more revenue, instead of embedding smaller requests just to increase the acceptance rate.

The proposed VM-ACO algorithm obtains more average generated revenue than the greedy algorithm. The reason is that K candidate mappings for each request enlarge the solution space than only one mapping selected in the greedy algorithm.

Fig. 4 provides the average cost of a VN request when the arrival rate of VN requests changes from 4 to 8 per 100 time units.

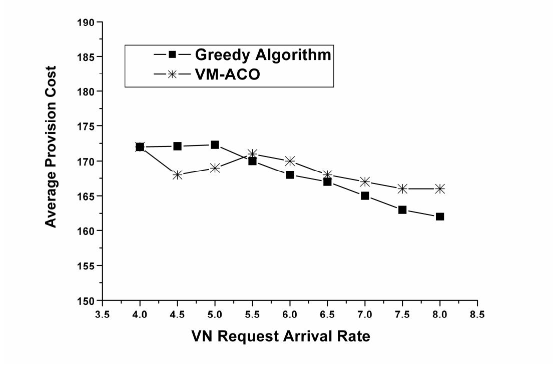

Fig. 4: Average provisioning cost with different VN arrival rates

As fig. 4 shows, the proposed VM-ACO algorithm achieves a similar provisioning cost as the greedy algorithm at different arrival rates for VN requests. In some cases, the provision cost of the proposed algorithm is a little larger than that of the greedy algorithm, because the proposed algorithm seeks a higher acceptance ratio by selecting a candidate network with some slightly longer SN paths. More revenue along with a little longer paths reflects the trade-off between the resource utilization and shortestness of provision paths [15].

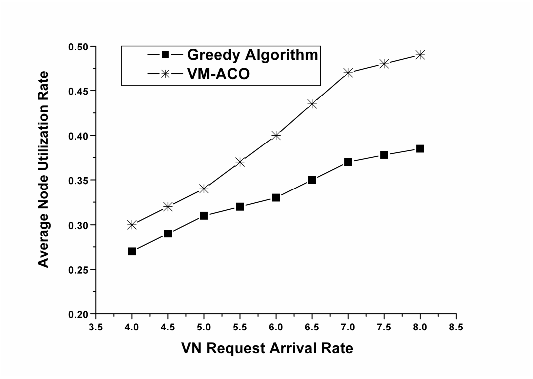

Fig. 5: Average node utilization rate with different VN arrival rates

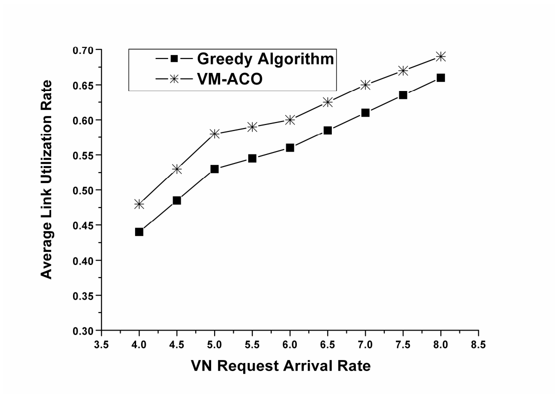

Fig. 6: Average link utilization rate with different VN arrival rates

To evaluate the resource utilization, fig. 5 and fig. 6 give the average node utilization rate and the average link utilization rate respectively when the arrival rate of VN requests change from 4 to 8 per 100 time units.

As fig. 5 and fig. 6 show, the average node and link utilization rate have the same tendency of becoming higher with more VN requests. This is because that more VN were accepted when the VN arrival rate changed from 4 to 8 per 100 time units. Moreover, the average node and link utilization rate of our proposed VM-ACO algorithm are both higher than ones of the greedy algorithm. The reason is our proposed algorithm embedded more requests than the greedy algorithm.

There are differences between the node and link utilization rate: the node utilization rate increased rapidly and the link utilization rate increased slowly. This indicates that the nodes of the SN are used more efficiently. The reason is that in the greedy algorithm more virtual links shared the same physical links. The physical nodes connecting them lost the capacity of embedding other requests. Because our proposed algorithm has K candidate networks for each request and has more chances to avoid the sheerness of virtual links, each SN node has more opportunities to embed more requests.

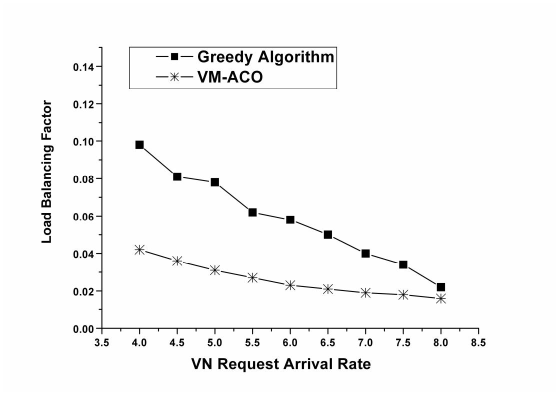

Fig. 7: Load balancing factor with different VN arrival rates

Fig. 7 provides the variance of resource utilization rate which is the combination of link and node utilization rate (load balancing factor defined in this paper) with different VN arrival rates. The variance is used to reflect the different usage of the nodes and links of the SN. As fig. 7 shows, the variance has lower values with more VN requests. This is true because that when more VN requests are needed to embed in the SN, there are more opportunities to gain the load balance.

Fig. 7 also shows that our proposed algorithm has lower variance values than the greedy algorithm. This is also because K candidate networks give the algorithm more chances to embed the VN requests efficiently.

CONCLUSION

In this paper, a batch process version of the VNE problem was defined based on the MCKP model. The goal of the batch process version of the problem is to take a given set of VN requests and embed the requests in the SN based on resource constraints in a way that maximizes the revenue of the InP. This is an NP-hard problem, for which we propose a modified ACO algorithm. In our algorithm, K candidate networks are first found for each request, and two types of pheromones are defined for a VN request and its candidate networks according to the characteristic of MCKP. To evaluate a solution, we introduce a load-balancing factor as a component of the fitness function and define the corresponding pheromone update rules. To overcome the initial blindness of the ACO algorithm, we define the cost of each candidate network as heuristic information. Our modified ACO algorithm provides better optimization performance than a greedy algorithm.

Acknowledgments
This study is supported by the Natural Science Foundation of Shandong Province (Grant No. ZR2013FM029) and University Innovation Project of Jinan (Grant No. 201303010).

REFERENCES

[1] T. Anderson; L. Peterson; S. Shenker, *Computer*, **2009**, vol. 38, 34-41.
[2] M. Chowdhury; M.R. Rahman; R. Boutaba, *IEEE/ACM Transactions on Networking*, **2012**, vol. 20, 206-219.
[3] G. Sun; H.F. Yu; V. Anand; L.M. Li, *Future Generation Computer Systems*, **2013**, vol. 29, 1265-1277.
[4] X.L. Li; H.M. Wang; B. Ding; X.Y. Li; D.W. Feng, *Future Generation Computer Systems*, **2014**, vol. 32, 1-12.
[5] G. Even; M. Medina; G. Schaffrath; S. Schmid, *Theoretical Computer Science*, **2013**, vol. 496, 184-194.
[6] J.F. Botero; M. Molina; X.H. Serra; J.R. Amazonas, *Journal of Network and Computer Applications*, **2013**, vol.36, 1735-1752.
[7] X. Cheng; S. Su, *Computer Networks*, **2012**, vol. 56, 1797-1813.
[8] S.D. Qing; J.X. Liao; X.M. Zhu; J.Y. Wang; Q. Qi, *Journal of Software (Chinese)*, **2012**, vol. 23, 3045-3058.
[9] Q. Zhu; H.Q. Wang; H.W. Lv; Z.D. Wang, *Journal on Communications*, **2012**, vol. 33, 170-177.
[10] D. Anderson, *Technical Report*, **2002**.

[11] M. Yu; Y. Yi; J. Rexford; M. Chiang, *Computer Communication Review*, **2008**, vol. 38, 17-29.
[12] Z.G. Ren; Z.R. Feng; A.M. Zhang, *Information Sciences*, **2012**, vol. 182, 15-29.
[13] M. Dorigo; V. Maniezzo; A. Colorni, *IEEE Transactions on Systems, Man, Cybernetics, Part B*, **1996**, vol. 26, 29-41.
[14] E. Zegura; K. Calvert; S. Bhattacharjee, *IEEE INFOCOM*, **1996**, 594-602.
[15] N. Wang, *IEEE Communications*, **2008**, vol. 10, 36-56.