Particle Swarm Optimization Based Multi-Domain Virtual Network Embedding

Kailing Guo, Ying Wang, Xuesong Qiu, Wenjing Li, Ailing Xiao
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China
Email: {klguo, wangy, xsqiu, wjli, xiao_ailing}@bupt.edu.cn

***Abstract*—Multi-domain virtual network embedding (MVNE) aims to embed a virtual network (VN) across multiple physical domains while minimizing the embedding cost. A key phrase of MVNE is VN partitioning which partitions a VN into multiple physical domains. Since the MVNE problem is NP-hard, we provide a heuristic VN partitioning approach named VNP-PSO based on the Particle Swarm Optimization (PSO) to increase the efficiency of VN partitioning. The VNP-PSO algorithm generates a near-optimal solution of VN partitioning through the evolution process of the particles. The simulation results show that our proposal can increase the efficiency of VN partitioning and decrease the embedding cost of MVNE.**

Keywords—multi-domain virtual network embedding; virtual network partitioning; Particle Swarm Optimization

I. INTRODUCTION

Network virtualization allows multiple virtual networks (VNs) to be embedded in a shared substrate network which may consist of one or more physical domains [1-6]. Each physical domain belongs to an infrastructure provider (InP).The service provider (SP) builds VNs by leasing virtual resources from InPs. Virtual network embedding (VNE) is to allocate physical nodes and paths to embed the virtual nodes and links of the VNs. Although it is possible for a small size VN to be fully embedded in a single InP, wide-area VNs always need to be embedded in multiple InPs.

Multi-domain virtual network embedding (MVNE) allows a VN to be embedded in multiple InPs, which creates the need of virtual network provider (VNP). VNP is explained in [7], which is a layer between SPs and InPs. VNP is responsible for accumulating the limited resource information disclosed by InPs. SP sends VNs to VNP. Then, VNP builds the VNs according to the information accumulated. Existing solutions for MVNE can be classified into two categories: the distributed and centralized solutions. The distributed approaches [3, 4] can respect the willingness of both SP and InPs. However, they will lead to extra network transmission cost. The centralized approach [5] based on the assumption that the SP has an abstract view of the InPs (i.e., without knowing the information of peering nodes), may result sub-optimal solutions. The other centralized approach [6] investigates the feasibility of MVNE with LID, and deals with the VN partitioning problem by using an exact algorithm which does not further consider the efficiency of the algorithm.

This work was supported by the National High Technology R&D Program of China (2013AA013502), the National Key Technology R&D Program (2012BAH35F02), and the Fundamental Research Funds for the Central Universities BUPT 2013RC1103.

In this paper, we provide a heuristic VN partitioning approach named VNP-PSO based on the Particle Swarm Optimization (PSO) [8] to increase the efficiency of VN partitioning. In VNP-PSO, there is a swarm of particles which are randomly initialized. Each particle has its position and velocity. The position of each particle is a possible VN partitioning solution. The velocity leads the particle to fly to a new position. The particles fly in the problem space by updating their positions and velocities. A near-optimal VN partitioning solution will be generated through the evolution process of the particles.

II. RELATED WORK

The MVNE problem has been studied in [3-6] which can be classified into two categories: the distributed [3, 4] and centralized [5, 6].

In [3], SP needs to know at least one InP to send the VN. An InP receives the VN and accepts a part of the VN. Then the InP forwards rest of the VN to other InPs to complete the whole VN. In [4], SP sends the VN to all the InPs. The InPs volunteer to bid for the VN sub-segment they can bear. Then, SP partitions the VN according to the quotations. The distributed approaches [3, 4] can respect the willingness of both SP and InPs. However, they will lead to extra network transmission cost and do not take into account the peering nodes information. Thus, the distributed approaches cannot acquire optimal embedding solutions.

In [5], VN partitioning is solved using both max-flow min-cut algorithms and linear programming techniques. The main work is to compare the exact and heuristic VN partitioning approaches in terms of VNE efficiency, which is based on a highly abstract view of the underlay (i.e., without knowing the information of peering nodes). The abstract view of the underlay may result sub-optimal solutions. In [6], the work takes into account the peering nodes information and investigates the feasibility of MVNE with LID. The VN partitioning problem is solved by using an exact algorithm. However, the algorithm does not further consider the efficiency of the MVNE.

Different from the above related works, we focus on the VN partitioning problem of MVNE with LID, and provide a heuristic algorithm named VNP-PSO to improve the efficiency of VN partitioning.

III. PROBLEM FORMULATION

A. Physical Network

A physical network is represented by an undirected weighted graph $G_s = (N_s, L_s, A_{N_s}, A_{L_s})$, where N_s is the set of physical nodes, L_s is the set of physical links, A_{N_s} is the attributes of the physical nodes , and A_{L_s} is the attributes of the physical links. The attributes of a physical node may include Operation System, location, CPU capacity and unit price. The attributes of a physical link include bandwidth and bandwidth price. Each physical network belongs to an InP, denoted by $B_{G_s}^{InP}$. Each InP has peering nodes through which InP can communicate with other InPs. The link between a pair of peering nodes is called peering link. Fig. 1 shows the peering nodes (*P1, P2, P3, P4, P5, P6*) and their connection relation.

B. Virtual Network Request

Existing literatures (e.g., [1], [2], [4], [5]) use an undirected weighted graph to denote a VN request. Topology-based VN embedding has two major flaws: unnecessary topology constrains and inconsistent internal topology [10]. And, some VN requests only consist of virtual nodes and the required traffic flows between some pairs of the nodes. Therefore, we denote a VN request by using a traffic matrix, referring to [6].

C. Multi-domain VN Embedding

The process of MVNE consists of the following four stages.

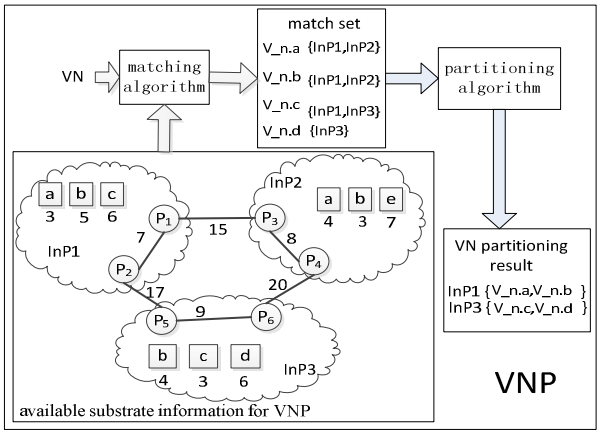

Fig. 1. the work flow of VNP

1) Resource information disclosing: In general, InPs disclose the static resource information, such as nodes types. InPs describe and advertise the non-confidential resource information which include nodes attributes (type, unit price) and peering links attributes (bandwidth price). VNP assembles and registers the disclosed information into a repository [5]. The price of the intra-domain links is confidential, therefore, we can only estimate the embedding cost by the price of the peering links and physical nodes price. In Fig. 1, the available substrate information for VNP include the set of node types (*a*, *b*, *c*, *d*, *e*) and nodes unit price, the set of peering nodes (P_1, P_2, P_3, P_4, P_5, P_6) and the link cost between them.

2) Resource matching: When a VN arrives, SP uses a matrix to denote the VN and then forwards it to VNP. VNP queries the repository and uses the matching algorithm [11] to find the match set $Match(n_v)$ for every virtual node n_v. The $Match(n_v)$ is the set of physical nodes suitable for the virtual node n_v to be embedded in. Since the intra-domain link price is confidential, the link match of the VN is unconsidered. In Fig. 1, the VN has four virtual nodes (*V_n.a*, *V_n.b*, *V_n.c*, *V_n.d*), the $Match(V\_n.a)$={InP1, InP2}, denoting that the physical node *a* in InP1 and InP2 are suitable for the virtual node $V\_n.a$.

3) VN partitioning: VN partitioning is an optimization problem. After the matching phase, VNP acquires the match set $Match(n_v)$ of every node $n_v \in N_v$. The physical nodes in $Match(n_v)$ may belong to different InPs. Therefore, VNP needs to partition the VN into multiple InPs based on the available information while minimizing the estimated embedding cost. As a result, VN is partitioned into several VN segments, where each segment is embedded in a particular InP. Fig. 1 illustrates a VN partitioning result. The virtual nodes *V_n.a* and *V_n.b* are partitioned into InP1, the virtual nodes *V_n.c* and *V_n.d* are partitioned into InP3.

4) Final VN embedding: After the VN partitioning phrase, VNP obtains a list of VN segments and sends them to corresponding InPs. Then, InPs embed their own VN segments into their physical networks [1, 2].

IV. VIRTUAL NETWORK PARTITIONING AMONG MULTIPLE INPS

VN partitioning is to partition a VN into multiple InPs, which is to find the suitable peering node for each virtual node while minimizing the embedding cost. We use a heuristic algorithm to resolve the VN partitioning problem.

A. The Discrete PSO for VN Partitioning

The embedding cost is defined as follows:

$$\text{cost} = \sum_{n_v \in N_v, n_s \in N_p} C(n_v, n_s) + \sum_{i,j \in N_v, m,n \in N_p} C(l_{ij}, p_{mn}) \quad (1)$$

N_v is the set of virtual nodes, and N_p the set of peering nodes. $C(n_v, n_s)$ denotes the cost of embedding virtual node n_v in peering node n_s. $C(l_{ij}, p_{mn})$ denotes the cost of embedding virtual link l_{ij}, *m* and *n* are the peering nodes that nodes *i* and *j* are embedded in, p_{mn} is the path that l_{ij} is embedded in. If *m* and *n* are two different peering nodes, $C(l_{ij}, p_{mn})$ equals to the peering link path cost between the two peering nodes, if *m* and *n* are the same node, the $C(l_{ij}, p_{mn})$ equals to 0. Thus, the link cost can be reflected onto the peering link cost. The first term of the function (1) is the total estimated cost of embedding all the virtual nodes. The second term of the function (1) is the total estimated cost of embedding all the virtual links.

This paper provides a heuristic VN partitioning approach named VNP-PSO based on the Particle Swarm Optimization (PSO) [8]. The variants of PSO are used for discrete optimization problem [9], which VNP-PSO will refer to. In VNP-PSO, there is a swarm of particles. Each particle has its position and velocity. The position of a particle is a possible solution. The velocity leads the particle to fly to a new position. The particles fly in the problem space by updating their positions and velocities. An appropriate-optimal solution of VN partitioning will be generated through the evolution process of the particles.

The position vector of the particle i is defined by $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$, x_{id} is the peering node that the dth virtual node is embedded in, and D is equal to the number of virtual nodes in the VN. The velocity vector of the particle i is defined by $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})$, v_{id} is a binary variable, if v_{id}=1, denoting the x_{id} is not suitable for the dth virtual node and should be replaced by another peering node which is associated with a physical node, the physical node is one of the match set of the dth virtual node; if v_{id}=0, remain the current position of the dth virtual node. In each iteration, after updating its position, the particle can calculate its fitness to know whether the new position is more suitable than before or not. The best position that the particle i has achieved denotes $P\_Best_i = (p_{i1}, p_{i2}, \dots, p_{iD})$. The best position that the swarm has achieved denotes $G\_Best = (g_1, g_2, \dots, g_D)$. The fitness function f is defined as follows.

$$f: \frac{1}{\sum_{n_v \in N_v, n_s \in N_p} C(n_v, n_s) + \sum_{i,j \in N_v, m,n \in N_p} C(l_{ij}, p_{mn})} \quad (2)$$

The denominator of the function f is the cost of the VN partitioning solution corresponding to the current particle. Therefore, the smaller the cost is, the bigger fitness value of the particle is.

The process of optimization problem can be regarded as the process of particles updating. The particle i updates its velocity and position according to (3) and (4).

$$v_{id} = \omega v_{id} + p_1(P\_Best_{id} - x_{id}) + p_2(G\_Best_d - x_{id}) \quad (3)$$

$$x_{id} = x_{id} * v_{id} \quad (4)$$

where v_{id} is the dth dimension of the velocity of particle i, x_{id} is the dth dimension of the position of particle i, ω is the inertia weight, p_1 is the cognition weight, and p_2 is the global weight. Set the values of ω, p_1, and p_2 to be constant values, where $\omega + p_1 + p_2 = 1$. P_Best_{id} is p_{id}, and G_Best_d is g_d.

The operations are defined as follows.

Operation "+": $p_i v_i + p_j v_j$ means that the particle keeps the velocity v_i with a probability p_i and keeps the velocity v_j with a probability p_j, where $p_i + p_j = 1$. For example,

0.7[1,0,1,0,1] + 0.3[1,1,0,0,1]=[1,# ,# ,0,1]

where "#" means that the value becomes 0 or 1 with the corresponding probability.

Operation "–": $x_{id} - x_{jd}$ denotes the difference between two position x_{id} and x_{jd}. If x_{id} and x_{jd} have the same value, return 0, otherwise, return 1. For example,

[2,5,7,2,6] – [3,5,6,5,7]=[1,0,1,1,1]

Operation "*": $x_{id} * v_{id}$ means that the particle i updates its position x_{id} with the v_{id}. If the v_{id}=1, denoting the x_{id} should be adjusted by another candidate node from the set *Match* (n_d), where the n_d denotes the dth virtual node; otherwise, x_{id} remains unchanged. The result is a new VN partitioning solution. For example, [2,5,7,2,6] * [0,1,1,0,0] means that the x_{i2} and x_{i3} should be adjusted, denoting that the second and third virtual node should be embedded in new suitable peering nodes.

B. VNP-PSO Algorithm Description

The description of VNP-PSO algorithm is as follows. The inputs of the algorithm are the information of a VN request and the virtual resource information that VNP accumulates. The output is a set of VN segments that are embedded in specific InPs. The fitness function f has been defined by function (2). If there isn't a peering link between two peering nodes, set the peering link cost to be ∞. If $f(X_i)$=0, denoting the particle i isn't a feasible VN partitioning solution.

VNP-PSO algorithm

1: Set the parameters.
2: Initial the particles. Randomly initialize the position vector X_i and the velocity vector V_i of the particles. Meanwhile Check that the particles are different from others.
3: Initial P_Best and G_Best. Set the vector P_Best_i to equal to the initial vector X_i; Calculate the fitness values $f(X)$ of all the particles according to the function (2), set the vector G_Best to equal to the vector X_j, where the value $f(X_j)$ of the particle j is the largest.
4: Update the velocity vector V and position vector X of the particles, according to the function (3) and (4).
5: Calculate the fitness values f(X) of all the particles. If $f(X_i)$ > $f(P\_Best_i)$, set X_i to be the P_Best_i, if $f(P\_Best_i)$ >$f(G\_Best)$, set P_Best_i to be the G_Best.
6: If the number of iterations is not equal to the pre-set value, go to step 4, otherwise go to step7.
7: Output the G_Best which denotes the best VN partitioning solution.

V. EVALUATION

In this section, we present the evaluation environment, the results and analysis of the simulation.

A. Evaluation Environment

The network topologies are randomly generated via GT-ITM tool [12]. The size of a physical network ranges from 50 to 100 nodes. The node capacity varies from 150 to 300. The unit price of a node capacity varies from 1 to 10. Each pair of nodes is connected by an intra-domain link with a probability of 0.5. The link capacity ranges from 1000 to 3000.There are two peering nodes in each InP. Each pair of peering nodes is connected by a peering link with a probability of 0.5. The bandwidth price of the peering link varies from 5 to 20. For a virtual network request, the number of virtual nodes varies from 5 to 15. The required capacity of a virtual node varies from 1 to 8 units. Each pair of virtual nodes is connected by a virtual link with a probability of 0.5. The required bandwidth of each virtual link varies according to a uniform distribution from 1 to 20.

The parameters in the algorithm are defined as follows. The number of the particles in VNP-PSO algorithm is set to be 15. The number of iterations is set to 60. The parameters ω, p_1 and p_2 are set to 0.2, 0.2 and 0.6, respectively. The best position G_Best of the swarm is a vital factor in the process of the evolution, therefore, it accounts for a lot of weight.

B. Comparison Method

The problem model we resolve is similar to [6] which proposes and investigates the problem of MVNE with LID. In [6], the VN partitioning is resolved by using an exact algorithm which does not further consider the efficiency of the algorithm. Therefore, we propose a heuristic algorithm named VNP-PSO to improve the efficiency of VN partitioning. We mainly compare our VN partitioning algorithm (VNP-PSO) with the exact VN partitioning algorithm (LID-Exact) [6].

C. Evaluation Results

We analyze the results from some aspects, including the VN partitioning time and embedding cost of the VNP-PSO algorithm. Detailed analysis is as follows. The partitioning time is the run time of partitioning a VN request by using the VN partitioning algorithm. The cost is the optimal cost of embedding a VN request denoted as function (1). We compare the efficiency of VNP-PSO and LID-Exact, in which three InPs participate in. And, we also investigate the stability of VNP-PSO algorithm.

In Fig. 2, as the number of virtual nodes increases, both the run time of VNP-PSO and LID-Exact increase. The run time of LID-Exact increases approximately exponentially, and the VNP-PSO increases linearly. The reason of the result is that LID-Exact calculates all of the VN partitioning solutions to find the optimal solution, while VNP-PSO generates the approximate optimal solution by following the local best solution in each iteration. When the number of virtual nodes in the VN is smaller than 7, the run time of the LID-Exact is shorter than that of VNP-PSO, otherwise, the run time of VNP-PSO is shorter. Therefore, the LID-Exact is suitable for small VN, while the VNP-PSO is suitable for large-scale VN. The cost of two algorithms both increase linearly, the VNP-PSO increases faster, because the result of LID-Exact is an exact solution, while the VNP-PSO is an approximate optimal solution. Thus, VNP-PSO performs well in the situation of partitioning large-scale VNs.

To investigate the stability of VNP-PSO, we change the number of the participating InPs. Different numbers of InPs denote different type of substrate networks. In the Fig. 3, as the number of the InPs increases, the run time and the cost of partitioning the same VN do not increase obviously. It indicates that the VNP-PSO is stable when the substrate network changes.

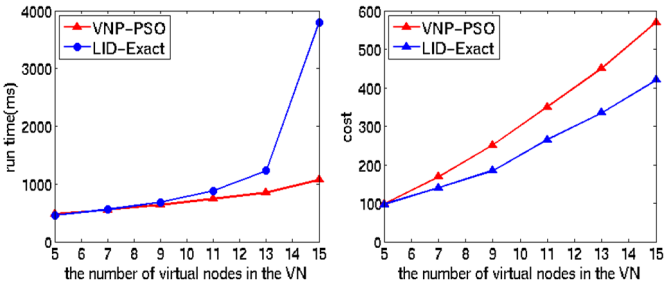

Fig. 2. Comparison of two VN partitioning algorithms

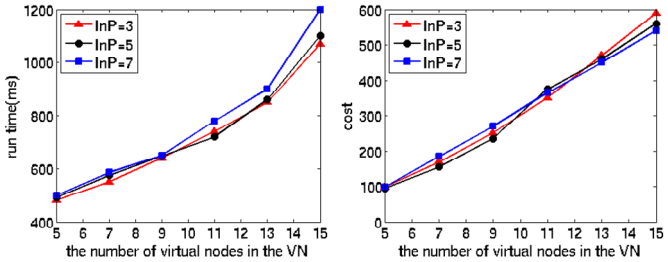

Fig. 3. The performance of VNP-PSO in the situations with different number of InPs

VI. Conclusion

Multi-domain virtual network embedding is one of the focuses in network virtualization. In this paper, we provide a heuristic approach named VNP-PSO based on the Particle Swarm Optimization (PSO) to increase the efficiency of VN partitioning. The simulation results show that the VNP-PSO algorithm is stable and suitable for large-scale VN. In the future, we plan to work in the two aspects: 1) design an efficient resource information management mechanism to improve the accuracy of resource matching results; 2) design an adaptive PSO algorithm which dynamically adjusts the parameters to further improve the efficiency of VN partitioning.

References

[1] M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual Network Embedding with Coordinated Node and Link Mapping," in 28th Conference on Computer Communications (IEEE INFOCOM), Rio de Janeiro,Brazil, April, 2009.

[2] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," Computer Communication Review, vol. 38, no. 2, pp. 17–29, 2008.

[3] Chowdhury M, Samuel F, Boutaba R., "PolyViNE: Policy-based virtual network embedding across multiple domains," ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures. 2010. 49-56.

[4] Zaheer F, Xiao J, Boutaba R, "Multi-Provider service negotiation and contracting in network virtualization," IEEE/IFIP Network Operations and Management Symp. 2010. 471-478.

[5] I. Houidi, W. Louati, W. Bean-Ameur, and D. Zeghlache, "Virtual Network Provisioning Across Multiple Substrate Networks," Computer Networks, 55(4), March 2011.

[6] Dietrich D, Rizk A, Papadimitriou P, "Multi-Domain Virtual Network Embedding with Limited Information Disclosure," IFIP Networking Conference. 2013. 1-9.

[7] G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldmann, R. Bless, A. Greenhalgh, A. Wundsam, M. Kind, O. Maennel, and L. Mathy, "Network Virtualization Architecture: Proposal and Initial Prototype," ACM SIGCOMM VISA, Barcelona, Spain, August 2009.

[8] J. Kennedy, R. Eberhart, et al., "Particle swarm optimization," IEEE International Conference on Neural Networks 4,1995, pp. 1942–1948.

[9] Xiang Cheng, Sen Su, Zhongbao Zhang, et al., "Virtual network embedding through topology awareness and optimization," Computer Networks, April 2012, pp. 1797-1813.

[10] C. Wang and T. Wolf, "Virtual Network Mapping with Traffic Matrices," ACM/IEEE ANCS, New York, USA, October 2011.

[11] H. Medhioub, I. Houidi, W. Louati, and D. Zeghlache, "Design, Implementation and Evaluation of Virtual Resource Description and Clustering Framework," IEEE AINA, Biopolis, Singapore, March 2011.

[12] GT-ITM. http://www.cc.gatech.edu/projects/gtitm/