Energy Aware Virtual Network Embedding

Sen Su Zhongbao Zhang* Alex X. Liu* Xiang Cheng Yiwen Wang Xinchao Zhao

***Abstract*—Virtual network embedding, which means mapping virtual networks requested by users to a shared substrate network maintained by an Internet Service Provider, is a key function that network virtualization needs to provide. Prior work on virtual network embedding has primarily focused on maximizing the the revenue of the Internet Service Provider and did not consider the energy cost in accommodating such requests. As energy cost is more than half of the operating cost of the substrate networks, while trying to accommodate more virtual network requests, minimizing energy cost is critical for infrastructure providers. In this work, we make the first effort towards energy aware virtual network embedding. We first propose an energy cost model and formulate the energy aware virtual network embedding problem as an integer linear programming problem. We then propose two efficient energy aware virtual network embedding algorithms: a heuristic based algorithm and a particle swarm optimization technique based algorithm. We implemented our algorithms in C++ and performed side-by-side comparison with prior algorithms. The simulation results show that our algorithms significantly reduce the energy cost by up to 50% over the existing algorithm for accommodating the same sequence of virtual network requests.**

***Index Terms*—Network virtualization, virtual network embedding**

I. Introduction

A. Background and Motivation

Network virtualization is the key technology that allows multiple heterogeneous Virtual Networks (VNs) to coexist on the same shared Substrate Network (SN). It brings three major benefits. First, it enables resource sharing among these VNs and makes most efficient use of the SN. Second, it offers opportunities to design and evaluate new network protocols and architectures. Third, it provides more flexibility to expand or shrink the VN as needed.

This paper concerns the problem of VN embedding. Network virtualization involves one Internet Service Provider (ISP) and multiple users, where the ISP manages the physical SN infrastructure while each user requests VNs from the ISP. Each VN request consists of a network topology where each node and edge have some constraints. The node constraints are typically on capacity (such as CPU computing power, memory and storage capacity, etc) and location (*i.e.*, the location in the topology of the substrate network of the ISP). The edge constraints are typically on communication bandwidth. When the ISP receives a VN requests from users, the ISP needs to map the VN to the physical nodes and links in its network, which is called VN embedding.

B. Limitation of Prior Art

The VN embedding has received significant attention in recent years. The primary goal of prior work is to maximize the revenue of the ISP by accommodating more VN requests on the same SN [2]–[8]. The key limitation of prior studies is that they did not consider the energy cost for serving VN requests. However, energy is a major cost for ISPs. For example, in US, Akamai, one of the world's leading providers of content delivery networking services, has an annual electricity cost of about $10 Million [9]. In China, China mobile Communications Corporation, the largest mobile service provider in the world, consumes over 13 TWH power consumption in 2011 [10]. Telecom Italia, the second largest consumer of electricity in Italia, consumes more than 2 TWh per year, which is equivalent to the energy consumed by 660,000 families in one year [11]. Thus, to maximize the net profit, the ISP needs to strike the right balance between accommodating more VN requests and minimizing energy costs for serving VN requests.

C. Proposed Approach

In this paper, we propose to tradeoff between maximizing the number of VNs that can be accommodated by an ISP and minimizing the energy cost of the whole system. For each VN request, the ISP maps the VN to some physical nodes and links in its network in such a way that the amount of additional energy cost caused by accommodating the VN request is minimized. This approach is based on two observations. The first observation is that the substrate nodes are usually geographically distributed to deploy and deliver service to end users, and the electricity price may differ for different locations and may fluctuate over time [9], [12]. Based on this observation, an ISP should try to map the virtual nodes of a VN to the physical nodes that have the lowest electricity price while satisfying the location constraint of the VN. The second observation is that the power consumption of a server is approximately in linear to its CPU utilization with a large offset, which equals up to nearly 50% of the peak power [13]. Based on this observation, an ISP should try to map the virtual nodes of a VN to the physical nodes that are already actively running; thus we can maximize the number of nodes that do not have any load and therefore can be put to sleep to save energy.

*Zhongbao Zhang and Alex X. Liu are the corresponding authors of this paper.

Some preliminary results of this paper were published in the First IEEE INFOCOM Workshop on Communications and Control for Sustainable Energy Systems: Green Networking and Smart Grids (INFOCOM WS-CCSES), Orlando, FL, March 25-30, 2012 [1].

Sen Su, Zhongbao Zhang, Xiang Cheng, Yiwen Wang are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. Email: {susen, zhongbaozb, chengxiang, wangyiwen}@bupt.edu.cn.

Alex X. Liu is with is with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA. E-mail: alexliu@cse.msu.edu.

Xinchao Zhao is with the School of Science, Beijing University of Posts and Telecommunications, Beijing, China. Email: xcmmrc@gmail.com.

D. Technical Challenges and Proposed Solutions

The first technical challenge is on modeling and quantifying the energy cost of the complex physical network infrastructure of an ISP. Specifically, we need to model both electricity price and energy consumption. For electricity price, we use a discrete-time model to characterize the spot dynamics of electricity price. For energy consumption, we first classify substrate nodes into *host nodes*, which need to execute some computational tasks, and *router nodes*, which need to forward packets to and from host nodes. We further classify them into *active nodes*, which need to be powered up, and *inactive nodes*, which can be powered off to save energy. Then, for different types of nodes, we build the corresponding energy consumption models. Based on such models, we carry out the quantitative analysis of the overall energy consumption, including the energy consumption of virtual nodes and virtual links for accommodating a VN request. After the modelings, we can quantify the electricity cost by calculating the time integral of the electricity price and the power consumption.

The second technical challenge is on designing energy aware VN embedding algorithms. To address this challenge, first, we model our energy aware VN embedding problem as an integer linear programming problem. Second, we propose a heuristic algorithm called *EA-VNE* for solving this problem. This algorithm consists of two steps: node mapping and link mapping. The node mapping step further consists of two substeps: router node mapping and host node mapping. In the router node mapping, we exploit the location- and time-varying diversities of electricity prices to save energy cost. To maximize the probability of performing successful link mapping in the next step, we design a worst-fit strategy for the bandwidth resources. In the host node mapping, we design a best-fit strategy to minimize the number of hosting nodes and make the best use of the resource while satisfying the node requirements of the VN request. In the link mapping step, we design an active nodes and router ports preferred shortest path algorithm that tries to minimize the number of forwarding nodes and ports. To further minimize energy cost, we design an approximation algorithm called *EA-VNE-EPSO*, which is based on the well known particle swarm optimization (PSO) technique. Specifically, we treat a VN embedding solution as a particle in PSO and thus each particle will achieve a better and better embedding solution through the iteration process by learning from the experience of other particles. To accelerate the convergence of this iterative algorithm, we propose an energy aware local selection strategy based on the characteristics of VN embedding. Furthermore, we propose a non-uniform mutation strategy to prevent premature convergence.

E. Summary of Experimental Results

We carry out extensive simulation and show that our algorithms outperform the state-of-the-art algorithm in terms of long-term average energy cost while gaining competitive revenues for ISPs. While maintaining nearly the same revenues, our algorithms EA-VNE and EA-VNE-EPSO save up to 40% and 50% of energy cost than prior art, respectively.

F. Key Contributions

We make the following key contributions in this paper:

1) We make the first attempt to incorporate the energy factor in performing VN embedding. We formulate an energy cost model for studying the energy aware VN embedding problem.
2) We design two VN embedding algorithms to reduce the energy cost while keeping nearly the same revenue so as to maximize the profit for the ISPs.
3) We conducted side-by-side comparison between our algorithms and the state-of-the-art algorithm. We show that our algorithms outperform the state-of-the-art algorithm in terms of both long-term average energy cost and revenues for ISPs.

The rest of the paper is organized as follows. In Section II, we present the energy cost model and the energy aware VN embedding problem formulation. In Section III and IV, we present our heuristic and meta-heuristic energy aware VN embedding algorithms, respectively. We evaluate our VN embedding algorithms in Section V. Section VI reviews related work. Finally, Section VII concludes the paper.

II. System Modeling

In this section, we first present a network model. Second, we formulate an energy model for VN infrastructure. Third, we quantitatively analyze the energy consumption for accommodating a VN request. Finally, we formulate the energy aware VN embedding problem based on the model. The notations used in this paper are summarized in Table I.

TABLE I: Notations

Notation	Description
s, t	Substrate nodes.
u, v	Virtual nodes.
$i(j)$	Substrate router (host) node.
$r(h)$	Virtual router (host) node.
$R_i(R_r)$	The residual (demanding) number of virtual routers.
$C_j(C_h)$	The residual (demanding) CPU value.
$M_j(M_h)$	The residual (demanding) memory value.
$S_j(S_h)$	The residual (demanding) storage value.
$B_{st}(B_{uv})$	The residual (demanding) bandwidth value.
$Dis(i, r)$	The Euclidean distance between r and i.
W	The maximum accepted distance value for mapping a virtual node to a substrate node.
$x_i^r(y_j^h)$	A binary variable. x_i^r represents router node mapping while y_j^h represents host node mapping. $x_i^r(y_j^h) = 1$ if mapping r (h) to i (j) and 0 otherwise.
f_{st}^{uv}	A binary variable. $f_{st}^{uv} = 1$ if mapping virtual link l_{uv} to the physical link l_{st} and 0 otherwise.
$PS_i(PS_j)$	A binary variable. $PS_i(PS_j) = 1$ if i (j) is in active state and 0 otherwise.

A. Network Modeling

A substrate network (SN) is represented by a weighted graph $G_s = (N_s, L_s)$, where N_s denotes the set of physical nodes and L_s denotes the set of physical links. The substrate nodes can be classified into two categories: router nodes and host nodes. That is, $N_s = (N_{sr}, N_{sh})$, where N_{sr} denotes the set of routers and N_{sh} denotes the set of hosts. Similarly, the substrate links can also be classified into two categories: the backbone link and the local link, denoted by L_{sr} and L_{sh}, respectively. Fig. 1 (b) shows an SN example where circles and rectangles denote router and host nodes, respectively.

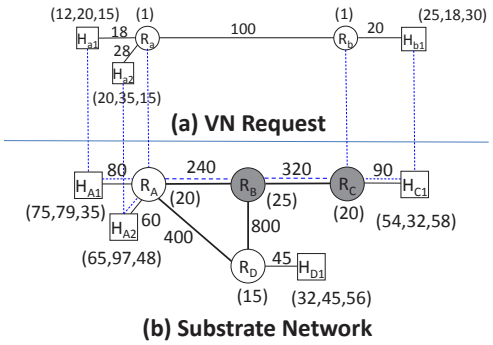


Fig. 1: Example of VN embedding

For router nodes, we consider the following three attributes. The first attribute is the number of virtual routers that the physical router can support for deploying and running different personalized network protocols. In Fig. 1 (b), the numbers that are in parentheses and besides each circle are such numbers. The second attribute is the location that the router is located as routers are generally geographically distributed. For example, in Fig. 1 (b), R_A may be located in Los Angeles, R_B in Chicago, R_C in New York, and R_D in New Jersey. We use a 2-dimensional coordinate $Loc(i) = (x_i, y_i)$ to denote the location of node i. The third attribute is electricity price. The power market of different locations are managed by different Independent System Operators (ISOs) and the ISOs are under competitive electricity market structure. Therefore, different locations often have different electricity prices. Even for the same location, the electricity price may vary frequently over time [9], [12]. Fig. 2 shows the hourly electricity price of the first week of Sep 2011 available at [14] for five regions in the day-ahead market, including Eastern Hub of PJM (Pennsylvania-Maryland-New Jersey), NP-15 Hub of CAISO (California), Capital Hub of NYISO (New York), Mass Hub of ISO-NE (New England) and Illinois Hub of MISO (Midwest). We observe that the electricity price varies over both location and time. To characterize the spot price dynamics, in this paper, we use a discrete time model, which has a time window (*e.g.*, an hour) of interest $t \in 0, 1, ..., T$. We use $Pr_i(t)$ to denote the electricity price for node i at time slot t, t_a to denote the arriving time of a VN request, and t_d to denote the duration of the VN request being served in the SN. Thus, the expiration time t_e of the VN request is $t_e = t_a + t_d$.

For host nodes, we consider three attributes: CPU speed, memory size, and storage capacity. In Fig. 1 (b), the triple besides each host node denotes the values of these attributes.

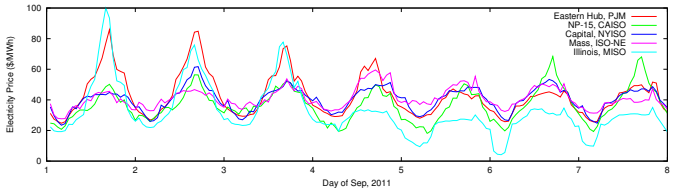


Fig. 2: Hourly electricity price of five locations in one week

For links, we consider bandwidth. In Fig. 1 (b), the number besides each link is the bandwidth. Note that the modeling, analysis, and algorithms in this paper can be easily extended to incorporate other attributes, such as latency and jitter constraints, as well.

A virtual network (VN) is represented as a weighted graph $G_v = (N_v, L_v)$. Here $N_v = (N_{vr}, N_{vh})$, where N_{vr} and N_{vh} denote the set of virtual router and host nodes, respectively; and $L_v = (L_{vr}, L_{vh})$, where L_{vr} denotes the set of virtual links between any two virtual routers and L_{vh} denotes the set of virtual links between host nodes and routers. Fig. 1 (a) shows the VN requested by a user on the SN in Fig. 1 (b).

We now formally define the VN embedding problem. Given a VN request G_v with a set of virtual nodes $N_v = (N_{vr}, N_{vh})$ and a set of virtual links $L_v = (L_{vr}, L_{vh})$, and a SN G_s with a set of physical nodes $N_s = (N_{sr}, N_{sh})$ and a set of physical links $L_s = (L_{sr}, L_{sh})$, embed G_v on G_s, which means to find two one-to-one mappings: M_n and M_l. Here M_n is a one-to-one mapping from N_v to a subset of N_s. This mapping includes two sub-mappings, M_{nr} and M_{nh}, where M_{nr} is from N_{vr} to a subset of N_{sr}, and M_{nh} is from N_{vh} to a subset of N_{sh}. For each virtual router node n_{vr} and the physical node $M_{nr}(n_{vr})$ that it maps to, $M_{nr}(n_{vr})$ satisfies both virtual router quantity and location requirements of n_{vr}. For each virtual host node n_{vh} and the physical node $M_{nh}(n_{vh})$ that it maps to, $M_{nh}(n_{vh})$ satisfies the node requirements on CPU speed, memory size, and storage capacity of n_{vh}. Here M_l is from L_v to a subset of P_s, which denotes all loop-free paths composed by the physical links in L_s. This mapping also includes two sub-mappings, M_{lr} and M_{lh}, where M_{lr} is from L_{vr} to a subset of P_{sr}, denoting all loop-free paths between any two routers, and M_{lh} is from L_{vh} to a subset of P_{sh}, denoting all loop-free paths between hosts and routers. For each virtual link l_v and the physical path $M_l(l_v)$ that it maps to, the bandwidth of each physical link in $M_l(l_v)$ is no less than the bandwidth requirement of l_v. Take the embedding in Fig. 1 as an example. The node mapping solution is $\{\{R_a \rightarrow R_A, R_b \rightarrow R_C\}, \{H_{a1} \rightarrow H_{A1}, H_{a2} \rightarrow H_{A2}, H_{b1} \rightarrow H_{C1}\}\}$ and the link mapping solution is $\{\{(R_a, R_b) \rightarrow (R_A, R_B, R_C)\}, \{(R_a, H_{a1}) \rightarrow (R_A, H_{A1}), (R_A, H_{a2}) \rightarrow (R_A, H_{A2}), (R_B, H_{b1}) \rightarrow (R_C, H_{C1})\}\}$.

Note that, we focus on considering one ISP in this paper. When multiple ISPs collaborate to provided VN services, as each ISP knows only the characteristics of his own infrastructure, there are many technical challenges to address this issue. The readers can refer to [15] for more discussion on this topic.

B. Energy Cost Modeling

1) ***Node Energy Cost****:* Let ΔPR_i^r denote the additional power consumption for mapping a virtual router $r \in N_{vr}$ to a substrate router $i \in N_{sr}$, and ΔPH_j^h denote the additional power for mapping a virtual host node $h \in N_{vh}$ to a substrate host node $j \in N_{sh}$. The node energy cost can be calculated as follows:

$$\Delta EN = \sum_{r \in N_{vr}} \sum_{i \in N_{sr}} x_i^r \Delta PR_i^r \int_{t_a}^{t_e} Pr_s(t)dt + \sum_{h \in N_{vh}} \sum_{j \in N_{sh}} y_j^h \Delta PH_j^h \int_{t_a}^{t_e} Pr_s(t)dt \quad (1)$$

This formula requires two inputs: ΔPR_i^r and ΔPH_j^h.

We first discuss the calculation of ΔPR_i^r. As a typical router usually consists of four main components as shown in

Table II, we estimate PR_i, the power consumption of mapping a router node, based on the model proposed in [16]:

$$PR_i = P_f + L \cdot P_l + P \cdot P_p. \tag{2}$$

Here L and P denote the number of linecards and ports of the routers, respectively. Thus, the incremental power consumption for mapping a virtual router node r to i is calculated as:

$$\Delta PR_i^r = \begin{cases} P_f + \Delta L \cdot P_l + \Delta P \cdot P_p \ (if \ PS_i \ = \ 0) \\ \Delta L \cdot P_l + \Delta P \cdot P_p \ (otherwise) \end{cases} \tag{3}$$

TABLE II: Components of a typical router

Components	Function	Power Notations
Chassis	Cooling equipments and others.	The sum of these two parts: P_f.
Switching fabric	Learning and maintaining the switching tables.	
Line-cards	Forwarding packets between the switching fabric and ports.	P_l
Ports	Transceiving packets in and out.	P_p

We now discuss the calculation of ΔPH_j^h. Many studies have reported that the full-system average power consumption of a typical server is approximately in linear with CPU utilization [17], [18]. The power consumption of other components, such as memory and storage, is very small [19]. We use the following equation to estimate the power consumption of mapping a host node:

$$PH_j = P_b + P_l \cdot util, \tag{4}$$

where P_b is the server's baseline power without any CPU load, and P_l represents the energy proportion factor for CPU utilization $util$. Thus, the additional power consumption for mapping a virtual host node h to j is:

$$\Delta PH_j^h = \begin{cases} P_b + P_l \cdot C_h \ (if \ PS_j \ = \ 0) \\ P_l \cdot C_h \ (otherwise) \end{cases}. \tag{5}$$

*2) **Link Energy Cost**:* We consider both long physical links, which span over a large geographical region and therefore require repeaters that consume power, and short physical links, which span a small geographical region and require no repeaters. We use PL_{st}^{uv} to denote the power consumption of the repeaters on a long link $l_{st} \in L_{sr}$ when mapping a virtual link $l_{uv} \in L_{vr}$. The overall link cost can be calculated as:

$$\Delta EL = \sum_{l_{uv} \in L_{vr}} \sum_{l_{st} \in L_{sr}} f_{st}^{uv} \Delta PL_{st}^{uv} Pr_s(t)dt \tag{6}$$

We set ΔPL_{st}^{uv} to be linear with the traffic volume of the l_{uv} and the distance between s and t based on the findings in [11]:

$$\Delta PL_{st}^{uv} = Dis(s,t) \cdot P_r \cdot \frac{B_{uv}}{OB_{st}}, \tag{7}$$

where P_r denotes the power density of the repeaters over distance and OB_{st} denotes the overall bandwidth capacity of substrate backbone link l_{st}.

*3) **Switching Cost**:* Powering up a router (or a server) incurs a onetime energy consumption for transiting from the power-saving state into the active state, which is called a switching cost. We use E_{sr} (or E_{sh}) to denote this cost. The overall switching cost can be calculated as:

$$\Delta ES = \sum_{r \in N_{vr}} \sum_{i \in N_{sr}} x_i^r \cdot (1 - PS_i) \cdot E_{sr} \cdot \int_{t_a}^{t_e} Pr_i(t)dt + \sum_{r \in N_{vh}} \sum_{j \in N_{sh}} y_j^h \cdot (1 - PS_j) \cdot E_{sh} \cdot \int_{t_a}^{t_e} Pr_j(t)dt. \tag{8}$$

C. Problem Statement of Energy Aware VN Embedding

With the goal of minimizing the overall energy cost $\Delta E = \Delta EN + \Delta EL + \Delta ES$ and the binary variables of x_i^r, y_j^h and f_{st}^{uv}, we next formulate the energy aware VN embedding problem as an integer linear programming (ILP):

Objective:

$$Min \quad \Delta E = \Delta EN + \Delta EL + \Delta ES \tag{9}$$

Capacity Constraints:

$$(\forall r \in N_{vr})(\forall i \in N_{sr}) : \begin{cases} x_i^r \cdot R_r \le R_i \\ x_i^r \cdot Dis(i,r) \le W \end{cases} \tag{10}$$

$$(\forall h \in N_{vh})(\forall j \in N_{sh}) : \begin{cases} y_j^h \cdot C_h \le C_j, \quad y_j^h \cdot M_h \le M_j \\ y_j^h \cdot S_h \le S_j, \quad y_j^h \cdot Dis(j,h) \le W \end{cases} \tag{11}$$

$$(\forall l_{st} \in L_s)(\forall l_{uv} \in L_v) : f_{st}^{uv} \cdot B_{uv} \le B_{st} \tag{12}$$

Connectivity Constraint:

$$(\forall s \in N_s)(\forall l_{uv} \in L_v) : \sum_{l_{st} \in L_s} f_{st}^{uv} - \sum_{l_{ts} \in L_s} f_{ts}^{uv} = \begin{cases} 1, & \text{if } x_s^u = 1 \text{ or } y_s^u = 1 \\ -1, & \text{if } x_s^v = 1 \text{ or } y_s^v = 1 \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

Variable Constraints:

$$(\forall i \in N_{sr}) : \sum_{r \in N_{vr}} x_i^r \le 1; \quad (\forall r \in N_{vr}) : \sum_{i \in N_{sr}} x_i^r = 1$$
$$(\forall j \in N_{sh}) : \sum_{h \in N_{vh}} y_j^h \le 1; \quad (\forall h \in N_{vh}) : \sum_{j \in N_{sh}} y_j^h = 1 \tag{14}$$

$$(\forall i \in N_{sr})(\forall r \in N_{vr}) : x_i^r \in \{0,1\},$$
$$(\forall j \in N_{sr})(\forall h \in N_{vh}) : y_j^h \in \{0,1\},$$
$$(\forall l_{st} \in L_s)(\forall l_{uv} \in L_v) : f_{st}^{uv} \in \{0,1\} \tag{15}$$

D. Performance Metrics

We define *the long-term average revenue*, earned by the ISP for accommodating VN requests, as follows:

$$\lim_{T\to\infty} \frac{\sum_{i=1}^{N} R^i(G_v)}{T}, \tag{16}$$

where $R^i(G_v) = (\sum_{h\in N_{vh}} (C_h + M_h + S_h) + W_R \cdot \sum_{r\in N_{vr}} R_r + \sum_{l_v\in L_v} B(l_v)) \cdot t_d$ represents the revenue for accommodating the ith VN request with the pay-as-you-go billing model. Here W_R denotes the market pricing weight of the resource of virtual routers over host node resources. To calculate energy cost in the long run, we define *the long-term average energy cost*, which the ISP must pay, as follows:

$$\lim_{T\to\infty} \frac{\sum_{i=1}^{N} E^i(G_v)}{T}, \tag{17}$$

where N is the number of VN requests accepted by the SN successfully in time T and $E^i(G_v)$ denotes the energy cost for the ith VN request. In this paper, we aim to both maximize the revenue and minimize the energy cost at the same time.

III. Energy Aware Heuristic VN Embedding Algorithm

Solving ILP is well known to be NP-hard [20]. Although standard exact algorithms such as branch and bound (BB) and cutting plane (CP) guarantee optimal solutions, they may incur exponential running time. Thus, they are not practical for online VN embedding when the problem size is large. In this section, we present a simple heuristic yet efficient algorithm, called EA-VNE, to produce an energy aware solution. EA-VNE is a two-step algorithm: the first step handles node mapping and the second handles link mapping.

A. Node Mapping

In this step, we first perform the router node mapping, then the host node mapping.

1) Router Node Mapping: We have two goals to achieve in this mapping. First, we want to optimize the energy cost in mapping virtual router nodes. Second, as router node mapping affects both host node mapping and link mapping, we want to perform router node mapping so that the host node mapping and link mapping will be successful and optimized.

Towards the first goal, we need to map the virtual router node on such substrate router nodes that have low electricity price and are in the *active* state. For each substrate router node i, we calculate the additional energy cost when we map the virtual router node r on it as follows: $E_i^r = \int_{t_a}^{t_e} Pr_i(t) PR_i^r dt$. We then sort the candidate substrate nodes according to the values of E_i^r in non-decreasing order. We use NR_E to denote the set of ranking values of substrate router nodes.

To achieve the second goal, we design a worst-fit scheme for the bandwidth constraint by selecting the substrate nodes with higher degree and bandwidth. This will make the subsequent host node mapping and link mapping easier.

Inspired by Google's Pagerank algorithm, we rank each substrate router node s based on itself and it's neighbors, called *Noderank*. We first measure the bandwidth resource of each substrate node s by $H(s) = \sum_{l\in L(s)} B_l$, where $L(s)$ denotes the set of all the outgoing links of s, and B_l denotes the available bandwidth resource of link l. Next, we compute the initial $N_R^{(0)}$ value for node s by $N^{(0)}(s) = \frac{H(s)}{\sum_{w\in N_{sr}} H(w)}$. In Pagerank, from current page, a random walker can follow it to another page that it links to and can also jump to any other page. We name these two operations as *forward* and *jump* operations. Let $s, t \in N_{sr}$ be two different nodes. Let $p_{st}^J = \frac{H(t)}{\sum_{w\in N_{sr}} H(w)}$, $p_{st}^F = \frac{H(t)}{\sum_{w\in nbr_1(s)} H(w)}$, where p_{st}^J denotes the jumping probability from node s to land on node t, $nbr_1(s) = \{t \mid (s,t) \in L_{sr}\}$, and p_{st}^F the forward probability from node s to node t. Clearly, $\forall s \in N_{sr} : \sum_{t\in N_{sr}} p_{st}^J = 1, \sum_{t\in nbr_1(s)} p_{st}^F = 1$. For any node $t \in N_{sr}$, let

$$NR^{(i+1)}(t) = \sum_{s\in V} p_{st}^J \cdot p_s^J \cdot NR^{(i)}(s) + \sum_{s\in nbr_1(t)} p_{st}^F \cdot p_s^F \cdot NR^{(i)}(s), \tag{18}$$

where $p_s^J + p_s^F = 1$, $p_s^J \geq 0$, $p_s^F \geq 0$, and $i = 0, 1, \cdots$. Noderank represents both the bandwidth resource and the topological attribute of substrate nodes. See details in our previous work [5]. Finally, we sort these nodes according to the values of Noderank in non-increasing order. We use NR_R to denote the set of ranking values for these nodes. We choose the substrate nodes with higher ranking.

Next, we prove that our algorithm converges.

Theorem 3.1: The NR algorithm eventually converges at a steady state.

Proof: Each value of $NR^{(i)}$ can be treated as a state of the Markov chain. First, the number of states is finite since the number of the substrate router nodes is limited. Second, the chain is irreducible since each node is strongly connected. Third, the chain is aperiodic since each node affects its own rank value. Thus, the theorem follows. ■

To tradeoff between the above two potentially conflicting goals, we design a comprehensive measurement for substrate nodes as follows:

$$NR = \alpha \cdot NR_E + (1-\alpha) \cdot NR_R, \tag{19}$$

where α denotes the ranking weight ($0 < \alpha < 1$). Thus, we choose the substrate node with the highest overall ranking for mapping virtual nodes.

The benefits of such ranking measure are two-fold. First, it can save energy cost by exploiting the diversity of electricity price and reducing the number of active router nodes. Second, it increases the possibility of accepting the VN requests and hence producing more revenues for ISPs.

2) Host Node Mapping: In this step, the goal is to search a substrate host node for each virtual host node while meeting their node constraints on CPU speed, memory size, and storage capacity. As host node mapping affects link mapping, besides satisfying the above node constraints, we also need to consider link mapping in this host node mapping step. By treating virtual nodes as items and substrate nodes as

bins, this problem becomes the multi-dimension bin packing problem. Thus, when a VN request arrives, we use a best-fit strategy to satisfy its resource constraints, which means that we prefer to map a virtual host node to the active substrate host node that has the most average utilization of the four resource constraints (namely CPU speed, memory size, storage capacity, and bandwidth) after the mapping. Specifically, we first define the average utilization for substrate host node j as: $\bar{u}_j = \frac{C_j+M_j+S_j+B_j}{4}$, where C_j, M_j, S_j, and B_j denote the CPU, memory, storage and bandwidth consumption after mapping h to j, respectively. After that, we define the standard deviation: $\delta_j = \sqrt{\frac{(C_j-\bar{u}_j)^2+(M_j-\bar{u}_j)^2+(D_j-\bar{u}_j)^2+(B_j-\bar{u}_j)^2}{4}}$. Finally, we select such node that has the least standard deviation for mapping h. Such strategy helps to strike a balance among the four kinds of resources and increase the resource utilization. It also helps to save energy by giving priority to the use of the nodes in active state without incurring the baseline power and switching cost.

The pseudocode of the above router and host node mapping algorithms is in Algorithm 1.

Algorithm 1: Energy Aware Router Node Mapping

Input: G_v, G_s.
Output: Energy aware node mapping solution.

/*Router node mapping*/
for *each virtual node n_{vr} to be mapped in N_{vr}* **do**
  Construct the candidate substrate router node list.
  Exclude the substrate nodes labeled $used$ from the list.
  Compute the NR_E, NR_R and NR values for all candidate nodes in G_{sr} using Equation 19.
  Apply the worst-fit strategy and map n_{vr} to the substrate node n_{sr} with highest ranking.
  if *node resource constraints not satisfied* **then**
    return RNODE_MAPPING_FAILED
  Label n_{sr} $used$.
/*Host node mapping*/
for *each virtual node n_{vh} to be mapped in N_{vh}* **do**
  Find the set of local host nodes of the corresponding substrate router node $M_{vr}(n_{vr})$, where n_{vr} denotes the access router node of n_{vh}.
  Exclude the substrate nodes labeled $used$ from the set.
  Apply the best-fit strategy to map n_{vh} to n_{sh} in this set.
  if *node resource constraints not satisfied* **then**
    return HNODE_MAPPING_FAILED
  Label n_{sh} $used$.
return NODE_MAPPING_SUCCESS

B. Link Mapping

To map a virtual link l_{uv}, we need to find a loop-free path between the hosting nodes h_u and h_v that satisfies the bandwidth constraint of l_{ij}. In finding such a path, we need to consider the link distance and the power states of substrate nodes. Towards this end, we design a weighted shortest path algorithm with the preference of active nodes and ports over inactive ones. Specifically, we first pre-calculate the weighted shortest path between each pair of substrate nodes using Floyd's algorithm. We then try to iteratively increase the length between h_u and h_v from the pre-calculated shortest length until that we find a set of paths that has the same weighted shortest paths and satisfy the bandwidth constraint of l_{uv}. To measure such paths, we introduce the weighted sum of the number of inactive nodes and inactive ports:

$$L = \beta N_f + (1-\beta)N_p, \tag{20}$$

where N_f and N_p denote the number of inactive forwarding nodes and inactive ports of a path, respectively. Finally, we select the path with the smallest weighted sum value to reduce energy cost. The pseudocode of the above node mapping algorithm is in Algorithm 2.

Algorithm 2: Energy Aware Link Mapping

Input: G_v, G_s, $MAXLEN$ (indicating maximum length that is acceptable), LEN (a matrix denoting the pre-calculated length of the weighted shortest path between each pair of substrate nodes).
Output: Energy aware link mapping solution.

for *each virtual link l_{ij} to be mapped in L_v* **do**
  Set $\triangle L$ to 0.
  while $LEN(h_i, h_j) + \triangle L \leq MAXLEN$ **do**
    Find a set of all paths $Set(P)$ with length $LEN(h_i, h_j)$ + $\triangle L$ in the SN.
    Sort these paths in $Set(P)$ according to $\beta N_f + (1-\beta)N_p$ in non-decreasing order.
    for *each path in $Set(P)$* **do**
      if *link resource constraint satisfied* **then**
        Record the link mapping solution for l_{ij}.
        Goto Step 1.
    Set $\triangle L$ to $\triangle L + 1$.
  return LINK_MAPPING_FAILED
return LINK_MAPPING_SUCCESS

C. Time Complexity Analysis

The NodeRanks for NR_E and NR_R can be computed in polynomial time in terms of $|G_s|$, $|G_v|$ and $\max\{1, -\log\epsilon\}$, where ϵ is a desired precision. The link mapping step can also be done in polynomial time in terms of $|G_s|$, $|G_v|$ and MAXLEN. Thus, EA-VNE is a polynomial-time algorithm.

IV. Energy Aware Meta-heuristic VN Embedding Algorithm

In this section, we present a meta-heuristic energy aware VN embedding algorithm.

A. Motivation

As mentioned in Section III, the VN embedding problem is NP-hard and is difficult to obtain the global optimal solution when the problem size is large. We propose population based optimization methods because they provide multiple solutions and allow us to choose the best from these solutions [21].

Among existing population-based optimization methods, Particle Swarm Optimization (PSO) [22], as an efficient and powerful tool, has been successfully applied to a wide range of optimization problems. It is inspired by the flocking behavior of birds in their food hunting. The process of solving the optimization problem is analogous to the food hunting process where the optimal solution corresponds to the food position. In PSO, each bird, called a particle, has its own position, represented by $X_i = (x_i^1, x_i^2, \cdots, x_i^D)$, where D denotes the dimensions of the solution space. To find the optimal solution, each particle interacts with each others in a certain way and adjusts its search direction iteratively with the velocity $V_i = (v_i^1, v_i^2, \cdots, v_i^D)$. The velocity of the i-th particle depends on three factors: its own current position (denoted by X_i), its own personal best previous experience (denoted by pB_i), and the group best experience (denoted by gB) of

all members. During the iterative process, the velocity and position of particle i on the d-th dimension are updated as follows:

$$v_i^d = wv_i^d + c_1 r_1^d(pB_i^d - x_i^d) + c_2 r_2^d(gB^d - x_i^d)), \quad (21)$$
$$x_i^d = x_i^d + v_i^d, \quad (22)$$

where w denotes the inertia weight, c_1 denotes the cognition weight, c_2 denotes the social weight, and r_1^d and r_2^d denote two random variables uniformly distributed in the range of [0,1] for the d-th dimension. The search mechanism of the elitist-based learning strategy in PSO helps it to achieve high efficiency. Furthermore, the required parameters of PSO are less than other heuristic algorithms, such as ant colony optimization [23]. Thus, we choose PSO instead of other meta-heuristic algorithms.

In operational network, when a VN request arrives, the ISP treats the position of each particle in PSO as a possible VN embedding node solution. First, the ISP initializes particles (*i.e.*, VN embedding node mapping solutions) randomly. Then, the ISP checks the validity of these VN embedding node solutions in terms of link mapping and determines the quality of these solutions by calculating the additional energy cost of them. Next, these particles learn from each other and update their positions to achieve a better position according to the local and global information. In such a way, finally, a near-optimal solution of VN embedding can be obtained through the evolution process of the particles.

There are three technical issues that need to be addressed in employing PSO for minimizing the energy consumption of embedding VN requests. First, as the standard PSO algorithm only deals with continuous optimization problems, it is not directly applicable to VN embedding, which is a discrete optimization problem. To address this issue, we introduce a variant of discrete PSO by redefining the parameters and operations of the particles in Section IV-B. Second, the randomness of PSO may generate infeasible solutions that violates the resource constraints; even if a feasible solution can be obtained, it may not be a good solution and consequently lead to slow convergence for the PSO algorithm. To address this issue, we present an energy aware local selection (EALS) strategy to initialize and update the positions of particles in Section IV-C, which helps to accelerate the convergence process. Third, the inherent premature convergence problem of PSO may become an obstacle to further optimize energy consumption. To address this issue, we propose the non-uniform mutation strategy for the gB particle in Section IV-D to address the premature convergence problem of PSO.

B. Discrete PSO for VN Embedding

We next present a discrete method for energy aware VN embedding. First, we label the virtual router nodes with $1, 2, 3, ...|VR|$ and substrate router nodes with $1, 2, 3, ...|SR|$, respectively. Second, we label the virtual host nodes connected to the i-th virtual router node with $i_1, i_2, ..., |VH|_i$ and the substrate host nodes connected to the i-th substrate router node with $i_1, i_2, ..., |SH|_i$, where $|VH|_i$ and $|SH|_i$ denote the number of edge host node of the i-th virtual and substrate router nodes, respectively. Third, we use the position of the i-th particle as the node mapping solution $X_i = (XR_i, XH_i)$ where $XR_i = (x_i^1, x_i^2, \cdots, x_i^{|R|})$ and $XH_i = (x_i^1, x_i^2, \cdots, x_i^{|H|})$ represent the router node and host node mapping solutions, respectively, and $|R|$ and $|H|$ denote the numbers of virtual router nodes and host nodes, respectively. That is, in XR_i, we will map the first virtual router node to the substrate router node labeled x_i^1, the second virtual router node to the substrate node labeled x_i^2 and so on for the rest of the virtual router nodes. The same rule also applies for XH_i. For the i-th particle, the velocity vector $V_i = (VR_i, VH_i)$, where $VR_i = (v_i^1, v_i^2, \cdots, v_i^{|R|})$ and $VH_i = (v_i^1, v_i^2, \cdots, v_i^{|H|})$, which guides the current VN embedding solution to achieve a better solution, is defined as a binary vector. If v_i^d = 1, then we use the current choice; otherwise, the corresponding virtual node mapping should be adjusted by selecting another substrate node from its candidate node list. After that, the velocity and position of particle i on the d-th dimension can be updated accordingly to the following velocity and position recurrence relations:

$$v_i^d = P_1 v_i^d \oplus P_2(pB_i^d \ominus x_i^d) \oplus P_3(gB^d \ominus x_i^d)), \quad (23)$$
$$x_i^d = x_i^d \otimes v_i^d, \quad (24)$$

where P_1 is the inertia weight, P_2 the cognition weight, and P_3 the social weight, satisfying $P_1 + P_2 + P_3 = 1$.

C. Energy Aware Local Selection (EALS) Strategy

In the basic PSO algorithm, during the evolutionary process, it is common to generate and update the position parameters of the particles randomly within the corresponding range with equal probability. However, the generated solutions may violate the node and link constraints and thus leads to slow convergence. To address this issue, we propose an energy aware local selection (EALS) strategy for position initialization and particle updating to achieve quick convergence. In the EALS strategy, to select the substrate route nodes, we first rank them according to Equation 19. The higher ranking the substrate router node i has, the higher possibility that the i-th node is selected in the corresponding dimension of a particle. For the substrate host nodes, we also sort them in terms of the standard deviation of resources in the non-decreasing order. Similarly, the higher ranking the substrate router node j has, the higher possibility that the j-th node is selected in the corresponding dimension of a particle. This strategy helps to produce feasible and energy-efficient candidate VN embedding solutions and therefore accelerate the convergence process.

D. Non-uniform Mutation (NUM) Strategy for the gB Particle

While achieving a fast convergence speed, the gB model of PSO has a high chance of getting stuck in local optima [24]. To address this issue, we use the mutation idea proposed in [25] to guide the flying of gB particle in order to keep the solution population of PSO diverse, to expand the exploration scope in the solution space, and to keep the elitist learning mechanism of PSO active. The PSO algorithm is highly

sensitive to the mutation probability. On one hand, higher mutation probability expands the search space and makes it possible to find better solutions; however, it causes large disturbance of the group evolution, which reduces the convergence speed. On the other hand, lower mutation probability makes the algorithm converge faster; however, the diversity of the population evolution reduces and the opportunity of falling into local optima increases. Thus, we use a non-uniform mutation (NUM) strategy designed for the gB particle [26]. The main idea is that gB particle performs a mutation operation with a certain possibility, which is calculated by the NUM function as $f(k,r) = \sigma * (1 - r^{(1-\frac{k}{K})^b})$, where σ, K, and b are constants. Here k and r are random numbers in the range $(0,1)$. Here r is used to non-monotonously decrease the mutation probability. Thus, this approach helps to reduce the likelihood of premature convergence and guides the searching toward the promising domain area. Our algorithm is enhanced by this strategy to enrich the population diversity as well as get more optimal solutions.

E. EA-VNE-EPSO Algorithm

The EA-VNE-EPSO algorithm takes the SN and a VN request as inputs, using Formula 9 as the fitness function $f(X)$, and outputs an energy aware VN embedding solution. The pseudocode of this algorithm is in Algorithm 3.

Regarding the time complexity, this algorithm consumes $M \cdot I$ times more running time than the heuristic $EA-VNE$. However, M and I are usually limited and can be adjusted easily. Our experimental results show that this algorithm is effective and efficient.

Algorithm 3: EA-VNE-EPSO Algorithm

Input: G_v, G_s, M (the population size) and I (the maximum iteration count);
Output: Energy efficient VN embedding solution.

```
Construct a candidate list of substrate nodes for each virtual node in the VN request.
Initialize M particles using the EALS strategy, which corresponds to node mapping.
Check the validity of these particles using Algorithm 2, which corresponds to link mapping.
Initialize pB and gB according to f(X).
while iteration count i < I do
    Update X and V for each particle.
    if f(X) ≠ +∞ then
        Use Equation (23) and (24) to update them with the EALS strategy.
    else
        Re-initialize its position vector X using the EALS strategy and its velocity vector V randomly.
    Update pB and gB.
    if f(X_i) < f(pB) then Set X_i as the personal best position of the particle i.
    if f(pB) < f(gB) then Set pB as the global best position.
    Calculate the mutation probability for the global best particle.
    if needing to perform a mutation then Reinitialize the position vector X of this particle should adopt the EALS strategy for and update gB.
    i++;
```

V. Experimental Results

A. Experimental Setup

SN Topology: Similar to most previous studies, we used the GT-ITM tool to generate the SN and VN topology [27]. The SN topology is configured to have 500 routers, following the same order of magnitude with the ISP configuration in [11]. These routers are geographically distributed in a (25×25) grid, which can be equally separated into 5 regions, representing 5 different electricity markets. For these 5 electricity markets, we apply the real-world electricity price traces of 5 RTOs in September 2011 in US that are publicly available at [14]. For each electricity market, the electricity price varies every hour, which is termed as a time window. In the SN, each pair of routers are connected with a probability of 0.5. For each router, the maximum number of virtual machine supported is a random number in $[3,128]$ based on Juniper router configurations [28]. According to the typical access ability of routers, each access router connects to 32, 64 or 128 host nodes randomly. All of the router and host nodes are in the power *inactive* state initially. For each host node in the SN, the capacities of all available CPU, memory, storage, and bandwidth of local links are uniformly distributed between 50 and 100. For backbone links between two routers, the bandwidth is set to $100\times$ of that for local links, which are the typical configurations in practice.

VN Topology: We varied the scales of VN requests in small, regular, and large sizes; specifically, the numbers of virtual router nodes are uniformly distributed between 2 and 10, between 10 and 50, and between 50 and 100, respectively. For all of these scales, we set the average VN connectivity to be 50%. Each access virtual router connects to 8, 16 or 24 host nodes, randomly. Following the similar configuration used in [6], all of the QoS requirements (*i.e.*, the CPU, memory, storage and bandwidth) of the virtual nodes and local virtual links are uniformly distributed between 0 and 50. Similar to the above SN configuration, for the backbone virtual links between two routers, the bandwidth is set to $100\times$ of local virtual links. Similar to most prior studies on VN embedding, the VN requests arrive according to a Poisson process with an average arrival rate of 3 VNs per time window (*i.e.*, an hour); each request has an exponentially distributed lifetime with an average of 10 time windows. In each simulation, there are about 720 time windows (about a month), which corresponds to about 2,000 VN requests on average. We ran ten random different instances with these settings and calculated the mean of the ten runs.

Other Parameter Settings: Similar to [7], we set the location parameter LP, which is the ratio of W (in Table I) to the grid size, to $1/3$. Similar to [16], we set P_f, P_l and P_p to 375W, 315W, and 3W, respectively. Similar to [11], [29], we set P_b, P_l defined in Section II-B to 165W, 1.5W per CPU unit, respectively, and set P_r to 100W per distance unit. Similar to [30], we set the switching costs for routers and hosts, *i.e.*, E_{sr} and E_{sh}, to the corresponding energy consumption at the maximum load of one hour. In calculating the long-term average revenue, we set W_R to 100. In EA-VNE, the ranking weight α in Formula 19 is set to 0.5 and $MAXLEN$ is set to 8. In PSO based algorithms, the population size is set to 10 and the maximum iteration count is set to 50. The parameters P_1, P_2, and P_3 in Equation 23 are set to 0.1, 0.2, and 0.7, respectively.

Comparison Setup: We implemented our algorithms in C++ and performed side-by-side comparison with prior 9 major algorithms in Table III. The evaluation proceeds in the

following five steps. In the first step (described in Section V-B), to evaluate the performance of our heuristic algorithm, we compared them with the state-of-the-art algorithm, *i.e.*, D-ViNE-SP proposed in [6]. D-ViNE-SP solves the programming model using relaxation and rounding techniques. It uses deterministic node mapping with the shortest path link mapping. To perform side-by-side comparison, we modified their algorithm slightly in order to support the embedding for two categories of nodes, *i.e.*, route nodes and host nodes. In addition, to evaluate the effectiveness of the packing schemes that we adopted in the route and node mapping steps, we also compared EA-VNE to the other three packing scheme combinations as shown in Table III. In the second step (described in Section V-C), we evaluated our PSO based algorithms by setting different parameters and compared it to the ACO based algorithm. In this step, we also investigated the performance improvement of the EALS strategy and NUM strategy throughout the comparisons among the algorithms EA-VNE-PSO, EA-VNE-ACO, and EA-VNE-EPSO. Next, we extend the investigation to evaluate the impact of the VN scale on the performance of these algorithms in the third step (described in Section V-D). Finally, to quantify how close our algorithm is to the optimal algorithm, the fourth step (described in Section V-E) compares our solution achieved by EA-VNE-EPSO with the optimal solution achieved by the standard ILP solver GNU Linear Programming Kit. All simulation experiments were performed on the server with Intel 3GHz dual-core CPU, 2GB memory, 160GB disk, and Linux 2.6 OS.

TABLE III: Comparison of VN embedding algorithms

Notation		Algorithm Description
Heu	EA-VNE	Our energy aware VN embedding algorithm.
	VNE-BB	Similar to EA-VNE, apply both best-fit scheme in router and node mapping steps.
	VNE-BW	Similar to EA-VNE, apply best and worst-fit in router and node mapping steps, respectively.
	VNE-WW	Similar to EA-VNE, apply both worst-fit scheme in router and node mapping steps.
Meta-heu	EA-VNE-PSO	The energy aware VN embedding algorithm based on PSO technique.
	EA-VNE-ACO	The energy aware VN embedding algorithm based on the ant colony optimization technique used in [31].
	EA-VNE-EPSO	The enhanced version of EA-VNE-PSO with EALS and NUM strategies.
Relaxed and rouding	D-ViNE-SP	The state-of-the-art VN embedding algorithm, proposed in [6].
Optimal	GLPK	It uses the standard ILP solver of GNU Linear Programming Kit (GLPK) [32] to solve the ILP model and generate the *optimal* energy aware VN embedding problem.

B. Energy Aware Heuristic VN Embedding Algorithm

This subsection investigates the performance of our EA-VNE algorithm in terms of the long-term average revenue defined in Formula 16, the long-term average energy cost defined in Formula 17, and the running time. The evaluations were carried out on the regular-sized VN scale. We report the comparison results (with 95% confidence interval) in Fig. 3.

1) Impact of Ranking Weight: For the value of α, in EA-VNE, we set it to be adaptively adjusted to the environment of network resource rather than a constant value. When the SN has abundant resources for serving the VN requests, which means that it is easy to satisfy the resource constraint, the factor of the NR_R should be neglected, and we have more opportunity to optimize the energy cost by consolidating VNs into the smaller number of substrate nodes. In this case, our primary goal is to optimize the energy cost and α should be set to near 0. When the SN is overloaded (*i.e.*, most of the substrate nodes have been powered up and the resource of the nodes and links is exhausted), our primary goal is to accommodate more VN requests. As the space for optimizing the energy consumption is also limited, α should be set to near 1. Therefore, based on this idea, we set the the value of α to be the dynamic utilization of the network resource of the SN.

In Fig. 3, we compare EA-VNE (with dynamically changing value of α) with other modifications (α is set to constant values, *i.e.*, 0.1, 0.5, 0.9). As shown in Fig. 5, the value of α in EA-VNE keeps between 0.3 and 0.4. We find that when α increases, both of the revenue and the energy cost drop. However, EA-VNE nearly achieves the best performance in terms of both long-term average revenue and energy cost. That is because the dynamic changing value for weight α helps it find a right balance between the revenue and energy cost.

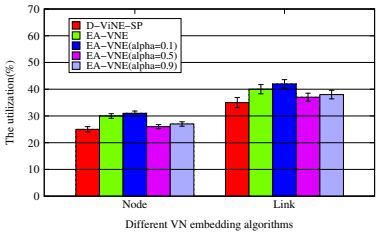
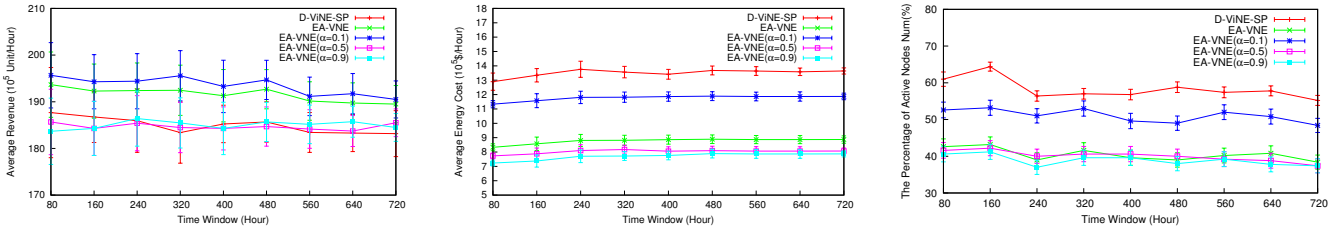


Fig. 5: The utilization

2) Comparison with State-of-the-art Algorithm: From Fig. 3(a), we observe that EA-VNE obtains a little more revenue than the state-of-the-art algorithm D-ViNE-SP. This is because first, in the node mapping step, in addition to node constraints, EA-VNE also considers the bandwidth constraint, which makes link mapping easier and therefore obtains higher possibility of accepting VN requests; second, VN consolidation increases the efficiency of the substrate resource usage and save more room for the incoming VN requests.

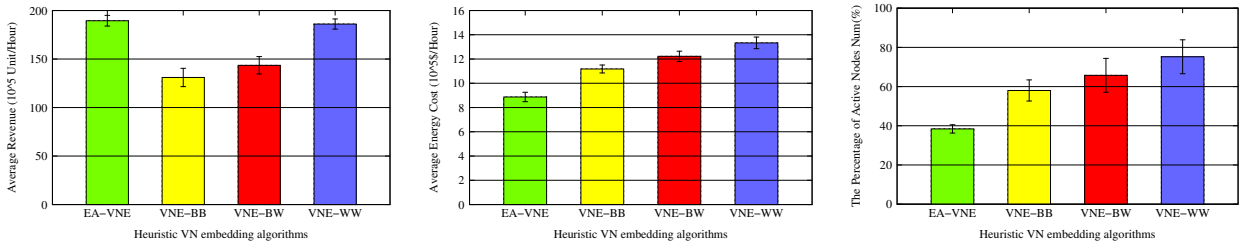
From Fig. 3(b), we observe that EA-VNE consumes much less energy than D-ViNE-SP in the long run. For example, at the time window of 720, the energy cost of EA-VNE is about 40% less than the energy cost of D-ViNE-SP. This is because EA-VNE considers the electricity price and performs consolidation in both the node and link mapping steps and avoid powering on inactive nodes. Thus, EA-VNE leads to higher energy efficiency than D-ViNE-SP. For example, Fig. 3(c) shows that EA-VNE consolidates the VN into much less number of substrate nodes by up to 30% than D-ViNE-SP.

Table IV shows the average running time. We observe that EA-VNE has much less running time than D-ViNE-SP.



(a) The long-term average revenue over time (b) The long-term average energy cost over time (c) The percentage of active nodes num over time

Fig. 3: Comparison between EA-VNE and the-state-of-the-art algorithm



(a) The long-term average revenue (b) The long-term average energy cost (c) The number of active substrate nodes

Fig. 4: Comparison among EA-VNE with different packing scheme combinations

This is mainly because D-ViNE-SP needs to solve two linear programming to embed a VN request [6].

TABLE IV: Average time and standard deviation comparison on regular-sized VN scale

Time	D-ViNE-SP	EA-VNE	VNE-BB	VNE-BW	VNE-WW
$\overline{T}$	1.20min	5.14ms	5.37ms	5.36 ms	5.36ms
δ_T	0.13min	0.60ms	0.35ms	0.35 ms	0.34ms

3) Comparison with Combinations of Other Packing Schemes: Table IV shows that EA-VNE consumes nearly the same running time as algorithms with combinations of other packing schemes, *i.e.*, VNE-BB, VNE-BW, and VNE-WW. Fig. 4 shows the comparison results with 95% confidence interval, from which we observe that EA-VNE has the following two benefits compared to these algorithms. First, EA-VNE generates the highest long-term average revenue with VNE-WW as shown in Fig. 4(a). For example, at the time window of 720, EA-VNE generates 45% and 32% higher revenue than VNE-BB and VNE-BW, respectively. VNE-BB and VNE-BW generate lower revenues because the best-fit strategy for router node mapping cannot guarantee a feasible link mapping solution and thus leads to a higher failure possibility for link mapping. Second, EA-VNE consumes lower long-term average energy than VNE-WW. Fig. 4(b) shows that EA-VNE reduces about 33% energy cost than VNE-WW. This is because the worst-fit scheme for satisfying the node constraints, used by VNE-WW, powers up more inactive substrate nodes as shown in Fig. 4(c) and thus incurs more energy cost.

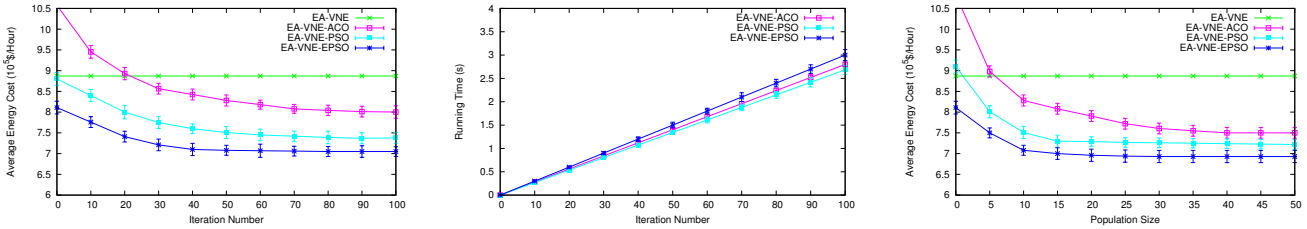
C. Energy Aware Meta-heuristic VN Embedding Algorithm

As mentioned in Section IV, each particle achieves a more energy-efficient solution through the iteration process by learning from the experience from other particles. As the number of iterations and population size increase, PSO can find better energy aware VN embedding solutions with the cost of more running time. To find the right balance between energy efficiency and running time, we carried out simulations on regular-sized VN scale to study the impact of the number of iterations and population size on these two performance metrics.

1) Impact of Iteration Number: We fixed the population size to 10 and varied the iteration number from 0 to 100. The energy efficiency and running time results are in Fig. 6(a) (with 95% confidence interval) and Fig. 6(b), respectively. The results show that for accommodating the same sequence of VN requests, PSO based meta-heuristic algorithms further reduces the energy cost within reasonable iteration number. When the iteration number is 50, EA-VNE-PSO can achieve 48% and 20% less energy cost than D-ViNE-SP and EA-VNE, respectively. When the iteration number is more than 50, as the iteration number increases, the running time of each our algorithm increases linearly as shown in Fig. 6(b), and the energy cost declines slowly and converges finally as shown in Fig. 6(a). Thus, we set the iteration number to 50 for our PSO based algorithm.

2) Impact of Population Size: In this evaluation, we set the iteration number to 50 and varied the population size from 0 to 50. Energy cost results are in Fig. 6(c). We observe that when the population size increases to more than 10, the energy cost declines slowly and finally converges. Thus, we set the population size to 10 in our PSO based algorithms. For running time, running time increases in linear with the population size for PSO. As the running time results are similar to those in Fig. 6(b), we do not show this figure due to space limitation. Note that PSO allows an ISP to set parameters (such as iteration number and population size) in tradeoff energy efficiency and running time.

3) Comparison to ACO technique: To evaluate the effectiveness of PSO, we compare EA-VNE-PSO to EA-VNE-ACO. For energy cost, Fig. 6(a) and 6(c) show that under the same iteration number and population size, EA-VNE-PSO



(a) The long-term average energy cost over iteration number

(b) The average running time over iteration number

(c) The long-term average energy cost over population size

Fig. 6: Comparison between EA-VNE-EPSO and other algorithms

outperforms EA-VNE-ACO significantly. For example, in Fig. 6(a), when the iteration numbers are 10 and 30, EA-VNE-PSO consumes about 12% and 10% less energy consumption than EA-VNE-ACO, respectively. Fig. 6(c) show similar results. There are two main reasons that EA-VNE-PSO outperforms EA-VNE-ACO significantly. First, after each iteration, the pheromone values in ACO are updated by all the ants that have built solutions [33]. Thus, ACO is less efficient than PSO. Second, as the pheromone accumulates, ACO stops searching early and thus leads to energy inefficiency and slow convergence, which can be seen from these two figures. For running time, Fig. 6(b) shows that these two algorithms are comparable.

4) Improvement of EA-VNE-EPSO over EA-VNE-PSO: Fig. 6(a) and Fig. 6(c), show that EA-VNE-EPSO has less energy cost than EA-VNE-PSO under the same iteration number and population size. For example, in Fig. 6(a), when the iteration numbers are 10 and 50, EA-VNE-EPSO saves about 8% and 6% less energy cost than EA-VNE-PSO. There are two main reasons that EA-VNE-EPSO has less energy cost. First, the EALS strategy accelerates the convergence of our algorithm. During the iterative process of PSO, EALS strategy helps to produce a larger number of VN embedding solutions, which are not only feasible but also energy-efficient. Second, the NUM strategy overcomes the premature phenomenon of PSO to some extent. The NUM strategy enhances the global search capacity of PSO and reduces its probability of being trapped into local optima. Thus, EA-VNE-EPSO achieves more energy-efficient VN embedding solutions and therefore saves more energy. Note that for the EALS strategy and its mutation process, EA-VNE-EPSO also consumes a little more running time than EA-VNE-PSO as shown in Fig. 6(b).

D. Impact of VN Scales

We now evaluate the impact of VN scales on the performance of our algorithms. As mentioned is Section V-A, we generated two more different sized VN inputs: small-sized and large-sized VN inputs. That means that the number of virtual router nodes is uniformly distributed between 2 and 10 and between 50 and 100, respectively. We conducted ten random different such instances for each type of scale. We report the comparison results (with 95% confidence interval) between the D-ViNE-SP, EA-VNE-ACO and our proposed algorithms in Fig. 7. From these results, we have the following interesting observations.

First, as the VN scale increases, all algorithms (*i.e.*, D-ViNE-SP, EA-VNE, EA-VNE-ACO, EA-VNE-EPSO) have the same rank in terms of energy cost and revenue.

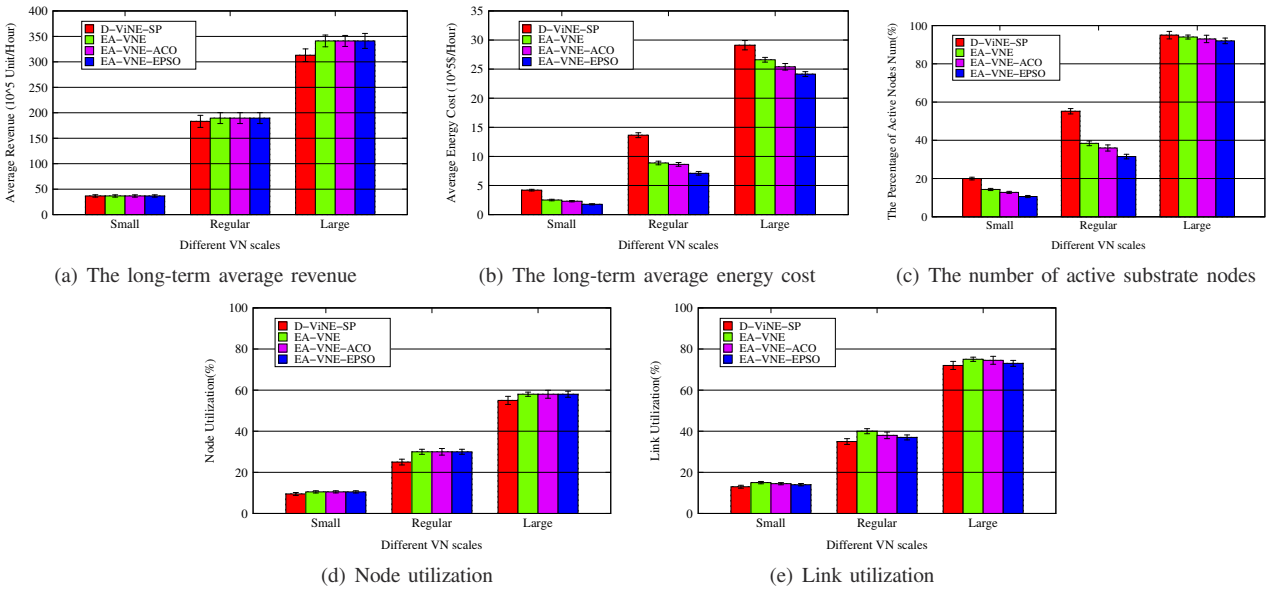
Second, from Fig. 7(b), as the VN scale increases, we observe that the relative energy efficiencies of our algorithms decline. For example, for small-sized VN requests, EA-VNE-EPSO (with the iteration number of 50) saves 23%, 29% and 58% energy cost than EA-VNE-ACO, EA-VNE and D-ViNE-SP, respectively. However, for regular-sized VN requests, EA-VNE-EPSO saves 17%, 19% and 47% energy cost than these algorithms. When the VN requests scale to large sizes, the relative energy efficiency advantages continue to drop to 4%, 9% and 16%, respectively.

As VN scale increases, Fig 7(d) and 7(e) show that all algorithms have higher node and link utilization; for each VN scale, they achieve nearly the same node and link utilization. However, with VN scale increasing, the difference in terms of the percentage of active nodes declines. This is because the SN has to power up more and more substrate nodes to satisfy the larger-sized VN requests as shown in Fig. 7(c). For the extreme case, when the VN infrastructure is overloaded, nearly all the substrate nodes have to be powered up to satisfy these VN requests. In such scenario, the powerdown based consolidation technique adopted in our algorithms may not work well. Thus, the energy saving efficiencies of our algorithms decline.

Third, from Table. V, which depicts the average running time and standard deviation comparison, we observe that as the VN scales from small sizes to large sizes, the average running time of these algorithms increase. It takes D-ViNE-SP some minute level on average to embed a VN request. This is because it needs to solve two linear programming to embed a VN request through relaxing and rounding techniques. However, the running times of our algorithms can keep at the several milliseconds or seconds level, which means that our algorithms are more practical for online VN embedding.

TABLE V: Running time comparison over different VN scales

Size	Time	D-ViNE-SP	EA-VNE	EA-VNE-ACO	EA-VNE-EPSO
Small	$\overline{T}$	0.10min	0.51ms	0.32s	0.34s
	δ_T	0.01min	0.04ms	0.02s	0.02s
Reg	$\overline{T}$	1.20min	5.14ms	3.06s	3.10s
	δ_T	0.13min	0.60ms	0.21s	0.23s
Large	$\overline{T}$	3.68min	13.02ms	8.02s	8.10s
	δ_T	0.44min	0.15ms	0.45s	0.42s



(a) The long-term average revenue (b) The long-term average energy cost (c) The number of active substrate nodes

(d) Node utilization (e) Link utilization

Fig. 7: Comparison between our algorithms and others on different VN scales

E. Optimality of EA-VNE-EPSO

To quantify the optimality of EA-VNE-EPSO, we compare EA-VNE-EPSO with the standard ILP solver GNU Linear Programming Kit (GLPK) [32]. Due to the inherent complexity of the optimal VN embedding problem, the time complexity of the ILP solver turns out to be exponential. We tried to map a VN request with 4 router nodes to the SN with 50 router nodes. It only took about several milliseconds for EA-VNE-EPSO to get a near-optimal solution while GLPK took even several days to compute the optimal solution, which is not practical for online VN embedding. Thus, for the comparison between EA-VNE-EPSO and GLPK, we especially carried out simulations on inputs of extra small-sized network topology, where only router nodes are included. Specifically, we set the number of virtual route nodes from 2 to 3 and varied the number of substrate nodes from 10 to 50. We embedded 500 such VN requests, all of which were accepted by the compared algorithm. T results in terms of the energy cost are in Fig. 8 (with 95% confidence interval).

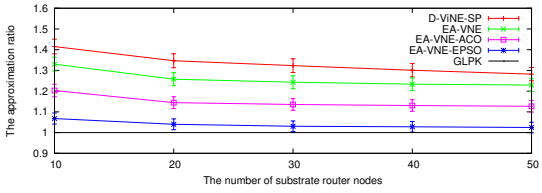


Fig. 8: The long-term average energy cost comparison

For running time, our results show that EA-VNE, EA-VNE-ACO, and EA-VNE-EPSO consume several milliseconds while D-ViNE-SP and GLPK consume several seconds on even such small network topologies. Fig. 8 shows the energy cost comparison. We observe that our algorithm EA-VNE-EPSO obtains the approximate energy efficient VN embedding solutions. For example, when the number of substrate nodes is 40, the approximation ratio of EA-VNE and EA-VNE-EPSO is about 1.233 and 1.028, respectively. When the number of substrate router nodes is 50, the approximation ratios of these two algorithms become 1.229 and 1.024, respectively. This indicates that our algorithms obtain not only the near optimal solutions but also have the stable performance.

We also evaluated the impact of location parameter LP on the approximation ratio of EA-VNE-EPSO by fixing the number of substrate nodes at 50 nodes. The results are in Fig. 9. We observe that as the LP increases, the approximation ratio of EA-VNE-EPSO decreases, which means that it achieves more optimal solutions. This is because larger LP indicates that EA-VNE-EPSO can explore larger space for exploiting the diversities of electricity price. For the extreme case, when $LP = 1$, it may place all the virtual nodes in the substrate nodes with the lowest price. Therefore, more optimal solutions can be achieved.

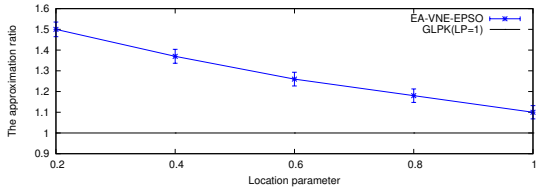


Fig. 9: The long-term average energy cost comparison

VI. Related Work

In recent years, the VN embedding problem has received much attention. However, most such work does not consider energy cost in carrying out VN embeddings.

Since VN embedding is a NP-hard problem, early studies made one or more of the following assumptions: (1) having the complete knowledge of VN requests [3]; (2) having neither node nor link requirements [34]; (3) having no admission control when the resource of the SN is insufficient [34]. However, these assumptions may not hold in reality.

Some attempts have been made to address VM embedding without the above assumptions [2], [5], [7], [8]; however, they introduce another assumption that the SN supports path splitting and migration. Path splitting means splitting a virtual link over multiple substrate paths to maximize revenue. Path migration means migrating the traffic on a path to another path to maximize the utilization of the SN. Yu *et al.* proposed the first VN embedding solution based on this assumption in [2]. Based on the assumption that path splitting is supported by the SN, later Cheng *et al.* proposed two VN embedding algorithms RW-Maxmatch and RW-BFS based on a topology-aware node resource measure called NodeRank, which reflects resource qualities; Zhang *et al.* designed a PSO based VN embedding algorithm to maximize the revenues from accommodating VN requests [8]; and Cheng *et al.* further leveraged NodeRank and PSO to maximize the revenues by trying to accommodate more VN requests [7].

Recently, also based on this assumption, Chowdhury *et al.* [6] considered the location requirement of virtual nodes and made the first attempt to formulate the VN embedding problem. To solve the formulation, they used the novel linear programming relaxation and rounding technique to coordinate the node and link mapping. Note that they applied multi-commodity flow (the shortest path) algorithm for the link mapping when the path splitting is (not) supported by the SN.

To minimize the physical resource of the SN, Fajjari *et al.* proposed a novel ant colony optimization (ACO) based VN embedding algorithm [31] called VNE-AC. It is inspired by the behavior of ants in finding best paths to launch an ant colony. In particular, each ant iteratively builds a piece of the solution (*i.e.*transition), where each solution component is embedded according to the available resources and the artificial pheromone trail in the SN. Compared to VNE-AC, our solution has the following advantages. First, after each iteration, the pheromone values in ACO are updated by all the ants that have built solutions [33]. Thus, the operations of ACO are less efficient than those of PSO. Second, the search mechanism of the elitist-based learning strategy of PSO ensures its efficiency. Besides, the key parameters of PSO are less than those of ACO [23]. Thus, it is easy to implement and modify in dealing with concrete problems. Another drawback of this algorithm is that, as the pheromone accumulates, it may not get a global optimum because it stops searching early [33].

All above work does not consider energy consumption. Since we publish the preliminary work of this paper, Botero *et al.* proposed an exact algorithm to solve the energy efficient VN embedding algorithm [35]. However, as demonstrated in Section V-E, applying exact algorithms (such as GLPK) to minimize energy cost will lead to too high time complexity and is not practical for online VN embedding. In contrast, in this paper, we focus on designing heuristic and meta-heuristic algorithms to achieve near-optimal VN embedding solutions with low calculational cost.

Some efforts are on addressing the inter-domain VN embedding problem, where VNs are provisioned across large-scale geographically distributed domains to deploy and deliver services end to end. Chowdhury *et al.* presented a decentralized inter-domain VN embedding framework [15]. In order to maximize the revenue for ISPs, Houidi *et al.* further designed an efficient inter-domain VN embedding algorithm, which first employed Max-flow/Min-cut approach to split the VN request into sub-requests and then used exact mapping to process such sub-requests [36] . These two studies also ignore the energy issue in VN embedding across multiple domains. While Zhang *et al.* [37] studied the cost aware VN embedding problem across multi domains, however, the network model is simple and the algorithm is also less efficient.

Recently, Seetharaman proposed to conserve energy by allowing tenants to reshape their own workload from the application aspect in a network virtualization environment [38]. However, our goal is to reduce energy cost from the resource scheduling aspect. Thus, it is orthogonal to our work.

VII. Conclusion

VN embedding is a key problem in network virtualization. In this paper, we study this problem from the energy saving perspective. Specifically, we propose an energy cost model and formulate the energy aware VN embedding to an integer linear programming. To solve this formulation, we design two energy aware VN embedding algorithms, *i.e.*, EA-VNE and EA-VNE-EPSO. Experimental results show that our algorithms can save up to 50% energy cost for ISPs than existing algorithms when accommodate the same sequence of VN requests.

Acknowledgment

The work was supported in part by National Natural Science Foundation of China under grants 61170274, the National Basic Research Program (973 Program) under grant 2011CB302704, National Natural Science Foundation of China under grants 61370226, 61272546, 61375066 and 61105127, the Innovative Research Groups of the National Natural Science Foundation under grant 61121061, the Important National Science & Technology Specific Projects - Research about Architecture of Mobile Internet (2011ZX03002-001-01), and China Scholarship Council. This work was also supported in part by the National Science Foundation under grants CNS-1017598, CNS-1017588, and CNS-1318563.

References

[1] S. Su, Z. Zhang, X. Cheng, Y. Wang, Y. Luo, and J. Wang, "Energy-aware virtual network embedding through consolidation," in *IEEE INFOCOM WS-CCSES: Green Networking and Smart Grids*, 2012, pp. 2708–2713.

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

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

[4] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, 2009, pp. 81–88.

[5] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual Network Embedding Through Topology-Aware Node Ranking," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 39–47, 2011.

[6] N. Chowdhury, M. Rahman, and R. Boutaba, "ViNEYard: Virtual Network Embedding Algorithms with Coordinated Node and Link Mapping," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 206–219, 2012.

[7] X. Cheng, S. Su, Z. Zhang, K. Shuang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology awareness and optimization," *Computer Networks*, vol. 56, no. 6, pp. 1797–1813, 2012.

[8] Z. Zhang, X. Cheng, S. Su, Y. Wang, K. Shuang, and Y. Luo, "A unified enhanced particle swarm optimization-based virtual network embedding algorithm," *International Journal of Communication Systems*, 2012.

[9] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for internet-scale systems," in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*. ACM, 2009, pp. 123–134.

[10] China Mobile Research Institute. [Online]. Available: http://www.greentouch.org/uploads/documents/Chih-Lin_I%20-%20TIA%20Green%20from%20a%20Service%20Provider%20Perspective.pdf

[11] L. Chiaraviglio, M. Mellia, and F. Neri, "Minimizing isp network energy cost: formulation and solutions," *IEEE/ACM Transactions on Networking (TON)*, vol. 20, no. 2, pp. 463–476, 2012.

[12] L. Rao, X. Liu, and W. Liu, "Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment," in *INFOCOM, Proceedings IEEE*, 2010, pp. 1–9.

[13] L. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[14] United States Federal Energy Regulatory Commission. [Online]. Available: www.ferc.gov

[15] M. Chowdhury, F. Samuel, and R. Boutaba, "Polyvine: policy-based virtual network embedding across multiple domains," in *Proceedings of the second ACM SIGCOMM workshop on VISA*. ACM, 2010, pp. 49–56.

[16] G. Ananthanarayanan and R. Katz, "Greening the switch," *Proceedings of HotPower*, 2008.

[17] S. Rivoire, P. Ranganathan, and C. Kozyrakis, "A comparison of high-level full-system power models," in *Proceedings of the 2008 conference on Power aware computing and systems*. USENIX Association, 2008, pp. 3–3.

[18] X. Fan, W. Weber, and L. Barroso, "Power provisioning for a warehouse-sized computer," in *Proceedings of the 34th annual international symposium on Computer architecture*. ACM, 2007, pp. 13–23.

[19] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan, "Full-system power analysis and modeling for server environments," in *In Proceedings of Workshop on Modeling, Benchmarking, and Simulation*, 2006, pp. 70–77.

[20] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons Inc, 1998.

[21] M. M. de Oca, T. Stutzle, K. Van den Enden, and M. Dorigo, "Incremental social learning in particle swarms," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 41, no. 2, pp. 368–384, 2011.

[22] J. Kennedy, R. Eberhart *et al.*, "Particle swarm optimization," in *Proceedings of IEEE international conference on neural networks*, vol. 4, 1995, pp. 1942–1948.

[23] W. Du and B. Li, "Multi-strategy ensemble particle swarm optimization for dynamic optimization," *Information Sciences*, vol. 178, no. 15, pp. 3096–3109, 2008.

[24] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: An overview," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–58, 2007.

[25] N. Higashi and H. Iba, "Particle swarm optimization with Gaussian mutation," in *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*. IEEE, 2003, pp. 72–79.

[26] X. Zhao, X. Gao, and Z. Hu, "Evolutionary programming based on non-uniform mutation," *Applied Mathematics and Computation*, vol. 192, no. 1, pp. 1–11, 2007.

[27] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *INFOCOM'96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*, vol. 2. IEEE, 1996, pp. 594–602.

[28] Juniper. [Online]. Available: http://www.juniper.net/us/en/local/pdf/datasheets/1000281-en.pdf

[29] L. Barroso and U. Hölzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synthesis Lectures on Computer Architecture*, vol. 4, no. 1, pp. 1–108, 2009.

[30] P. Bodik, M. Armbrust, K. Canini, A. Fox, M. Jordan, and D. Patterson, "A case for adaptive datacenters to conserve energy and improve reliability," 2008.

[31] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "Vne-ac: Virtual network embedding algorithm based on ant colony metaheuristic," in *Communications (ICC), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–6.

[32] "GNU Linear Programming Kit." [Online]. Available: http://www.gnu.org/software/glpk/

[33] Z. Wu, N. Zhao, G. Ren, and T. Quan, "Population declining ant colony optimization algorithm and its applications," *Expert Systems with Applications*, vol. 36, no. 3, pp. 6276–6281, 2009.

[34] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *INFOCOM*, 2006.

[35] J. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, and H. De Meer, "Energy efficient virtual network embedding," *Communications Letters, IEEE*, vol. 16, no. 5, pp. 756–759, 2012.

[36] I. Houidi, W. Louati, W. Ameur, and D. Zeghlache, "Virtual network provisioning across multiple substrate networks," *Computer Networks*, vol. 54, no. 4, pp. 1011–1023, 2011.

[37] Z. Zhang, S. Su, X. Niu, J. Ma, X. Cheng, and K. Shuang, "Minimizing electricity cost in geographical virtual network embedding," in *IEEE GLOBECOM*, 2012, pp. 2609–2614.

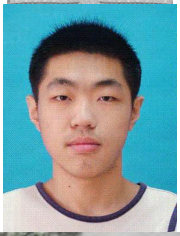
[38] S. Seetharaman, "Energy conservation in multi-tenant networks through power virtualization," in *Proceedings of the international conference on Power aware computing and systems*, 2010, pp. 1–8.

Sen Su received the Ph.D. Degree in Computer Science from the University of Electronic Science and Technology, China in 1998. He is currently a Professor of the Beijing University of Posts and Telecommunications. His research interests include distributed computing systems and the architecture of Internet services.

Zhongbao Zhang is currently a Ph.D. Candidate at Beijing University of Posts and Telecommunications in China. He visited Michigan State University as a joint PhD student from Sep 2012 to Sep 2013. His major is Computer Science. His research interests include networking and security.

Alex X. Liu received his Ph.D. degree in computer science from the University of Texas at Austin in 2006. He received the IEEE & IFIP William C. Carter Award in 2004 and an NSF CAREER award in 2009. He received the Withrow Distinguished Scholar Award in 2011 at Michigan State University. He is an Associate Editor of IEEE/ACM Transactions on Networking. He received Best Paper Awards from ICNP-2012, SRDS-2012, and LISA-2010. His research interests focus on networking and security.

Xiang Cheng is currently a Ph.D. Candidate at Beijing University of Posts and Telecommunications in China. His major is Computer Science. His research interests include network support for cloud computing and network virtualization.



Yiwen Wang is currently a PhD candidate from Beijing University of Posts and Telecommunications in China. His major is Computer Science. His research interests include network support for cloud computing and network virtualization.

XinChao Zhao is currently an Associate Professor of Beijing University of Posts and Telecommunications. His research interests include swarm Intelligence and evolutionary computing.