Energy aware virtual network embedding with dynamic demands: Online and offline

Zhongbao Zhang, Sen Su, Junchi Zhang, Kai Shuang, Peng Xu
State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China

Abstract

In Network as a Service model in cloud computing, how to efficiently embed virtual networks with both node and link demands into a shared physical network, namely virtual network embedding, has attracted significant attention. Most of prior studies on this problem have the following two limitations: (i) they assumed that the virtual network demands are constants, which does not hold in real-world network since such demands may vary a lot over time; (ii) their primary goal was to achieve more revenues for the physical network, with no consideration of the energy cost, which has become a more and more critical issue. In this paper, we bridge the gaps and study the energy aware virtual network embedding problem with dynamic demands. Specifically, we first model the dynamics of virtual network demands as a combination of a Gaussian distribution and a daily diurnal pattern. We then design two efficient heuristic algorithms by leveraging the dynamic characteristic of virtual network demands to minimize the energy consumption while keeping a high revenue for the physical network. One algorithm processes the virtual network requests one by one while the other one processes them group by group. We implemented these two algorithms in C++ and performed side-by-side comparisons with the prior algorithm. Extensive simulations show that our algorithms significantly reduce the energy cost by up to 25% over the state-of-the-art algorithm, while maintaining near the same revenue.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

Cloud computing has emerged as a promising paradigm in several fields (e.g., business, engineering and health-
key benefits of network virtualization in cloud computing are three-fold. First, it enables resource sharing at the network-level to make most efficient use of the PN. Second, it is possible for the VN to expand or shrink as needed. Third, it offers a pay-as-you-go pricing model for the users for renting their personalized topology network.

Two categories of constraints are present in a VN: node constraints and link constraints. The node constraints are typically on capacities of nodes (such as CPU, memory and storage capacity) and the relative location in the topology of the physical network. The link constraints are typically on communication bandwidth and latency. When the InP receives a sequence of VN requests from SPs, the InP needs to map the virtual nodes and links of these VNs onto the physical nodes and links in the PN, which is well known as the VN embedding problem.

Recently, the VN embedding problem has received significant attention. The primary goal of prior studies [4–14] is to design efficient embedding methods to maximize the revenue for accommodating the VNs in the shared PN. However, they ignored the energy cost, which has become an increasingly important factor for the InP. In our previous studies [15,16], we proposed to minimize the energy consumption when conducting VN embedding. Similar to most of prior art, these two studies still consider that the demands of virtual nodes and virtual links are constant values. However, in real-world applications, these demands may vary a lot over time and thus this consumption does not hold. In this paper, we revisit the VN embedding problem from both of the aspects of energy and dynamic demands. However, to address such a problem, there will be two challenges involved.

1. The first one resides in the modeling: how to capture and model the variation of VN demands in VN embedding.
2. The second one resides in the algorithm design: how to leverage the dynamic characteristic of VN demands to minimize the energy cost for the InP.

To address the first challenge, following the real demand traces and related studies [14,17–20], we observe that the VN demands exhibit a daily diurnal pattern and different demands have peaks at different time. Therefore, we model the VN demands as a combination of the Gaussian distribution and the daily diurnal pattern in this paper.

To address the second challenge, we formulate the VN embedding problem with dynamic demands to an integer linear programming (ILP) and design two heuristic algorithms to solve this formulation. One algorithm processes the VN requests one by one while the other one processes them group by group. The first algorithm is called EAD-VNE, which has two steps. The first step is node mapping and the second step is link mapping. In the node mapping, we strike the resource balance of the PN, exploit the dynamic characteristic of VN demands and pre-consider the subsequent link mapping to minimize the node energy consumption while achieving high revenue. In the link mapping, we design a weighted shortest path algorithm with the preference of active nodes and opposite variation trends between the bandwidth demand and the bandwidth load. Based on EAD-VNE, we design the other algorithm called EAD-VNE-G. This algorithm queues the VN requests and processes them group by group. The benefit of this algorithm lies in that it can further leverage the dynamic characteristic of VN demands for different VN requests and thus help benefit the energy conservation when performing VN embedding.

Through extensive simulations, we show that our algorithms outperform the state-of-the-art algorithm [16] in terms of long-term average revenue and energy consumption. In particular, our proposed two algorithms reduce up to 16 and 25% energy consumption than the state-of-the-art algorithm, respectively, while keeping nearly the same revenue.

We make the following major contributions in this work:

1. To the best of our knowledge, we make the first attempt to study the energy aware VN embedding problem with dynamic demands.
2. We formulate the problem into an ILP and design two heuristic algorithms, i.e., EAD-VNE and EAD-VNE-G, to optimize the energy consumption while generating high revenues for the InP.
3. We conduct side-by-side comparisons between our algorithms and the state-of-the-art algorithm to demonstrate the energy efficiency of our algorithms.

We organize the remainder of this paper as follows. Section 2 presents the modeling and problem formulation. Sections 3 and 4 present our two proposed heuristic algorithms: one processes the VN requests one by one and the other one processes the VN requests group by group. Section 5 evaluates these two algorithms. Section 6 reviews the related work. Finally, Section 7 concludes this paper.

2. Modeling and formulation

In this section, we first model the network in Section 2.1 and energy consumption in Section 2.2, respectively. Then we will present the formulation of this problem in Section 2.3. Finally, we will give the performance metrics in Section 2.4. Before that, we first summarize the notations in Table 1, which will be used throughout this paper.

2.1. Network modeling

Physical Network (PN): We model a physical network as a weighted graph $G_p = (N_p, L_p)$, where $N_p$ denotes the set of physical nodes and $L_p$ denotes the set of physical links. For physical nodes, the attributes generally include CPU, memory, disk and relative location in the network. For physical links, the attributes generally include bandwidth and latency. Similar to studies [9,10,15,16], we mainly consider the CPU and location as the node attributes and bandwidth as the link attribute. For example, at the bottom of Fig. 1, (c) shows a PN where the numbers near the nodes and links denote the available CPU capacity and bandwidth capacity, respectively. Note that our modeling, analysis, and algorithm in this paper can be easily extended to incorporate other attributes.

Virtual Network (VN): A sequence of virtual networks (VNs) arrive and depart over time. Similarly to $G_p$, we model each VN as a weighted graph $G_v = (N_v, L_v, t_a, t_e)$. Differently, in $G_v$, we have $t_a$ and $t_e$ for denoting the arrival time and expiration time of the VN, respectively. In $N_v$, $C_u$ and $D_u$ denote the CPU demand and required location of the virtual node $u$. In $L_v$, $B_{uv}$ denotes the bandwidth demand for the
Table 1
Notations.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>The sequence # of the VN request.</td>
</tr>
<tr>
<td>$G_p$</td>
<td>The physical network.</td>
</tr>
<tr>
<td>$N_p$</td>
<td>The set of physical nodes.</td>
</tr>
<tr>
<td>$L_p$</td>
<td>The set of physical links.</td>
</tr>
<tr>
<td>$G_v$</td>
<td>The virtual network.</td>
</tr>
<tr>
<td>$N_v$</td>
<td>The set of virtual nodes.</td>
</tr>
<tr>
<td>$L_v$</td>
<td>The set of virtual links.</td>
</tr>
<tr>
<td>$t_a$</td>
<td>The arrival time for a virtual node.</td>
</tr>
<tr>
<td>$t_e$</td>
<td>The expiration time for a virtual node.</td>
</tr>
<tr>
<td>$u$</td>
<td>The virtual nodes.</td>
</tr>
<tr>
<td>$v$</td>
<td>The physical nodes.</td>
</tr>
<tr>
<td>$lpq$</td>
<td>The virtual link between virtual nodes $u$ and $v$.</td>
</tr>
<tr>
<td>$p$</td>
<td>A binary variable.</td>
</tr>
<tr>
<td>$q$</td>
<td>A binary variable.</td>
</tr>
<tr>
<td>$\mu_v$</td>
<td>The mean value of CPU requirement of virtual node $u$.</td>
</tr>
<tr>
<td>$\sigma_v$</td>
<td>The standard deviation of CPU requirement of virtual node $u$.</td>
</tr>
<tr>
<td>$\mu_{lpq}$</td>
<td>The bandwidth demand of virtual link $lpq$.</td>
</tr>
<tr>
<td>$\sigma_{lpq}$</td>
<td>The standard deviation of bandwidth requirement of virtual link $lpq$.</td>
</tr>
<tr>
<td>$CPU(p)$</td>
<td>The CPU capacity of physical node $p$.</td>
</tr>
<tr>
<td>$BW(lpq)$</td>
<td>The bandwidth capacity of physical link $lpq$.</td>
</tr>
<tr>
<td>$P_b$</td>
<td>The baseline power for a hosting node.</td>
</tr>
<tr>
<td>$P_i$</td>
<td>The power proportional factor for CPU utilization for a hosting node.</td>
</tr>
<tr>
<td>$NH(t)$</td>
<td>The number of active forwarding nodes at the time of $t$.</td>
</tr>
<tr>
<td>$NF(t)$</td>
<td>The number of active forwarding nodes at the time of $t$.</td>
</tr>
<tr>
<td>$Dis(D_p, D_v)$</td>
<td>The Euclidean distance between virtual node $u$ for the ith VN request and the physical node $p$.</td>
</tr>
<tr>
<td>$R_{lu}$</td>
<td>It indicates how far virtual node $u$ for the ith VN request can be placed from the ideal location $D_u$.</td>
</tr>
</tbody>
</table>

**Virtual Network Embedding:** Now we formally present the definition of VN embedding as follows: Given a VN $G_v$ and a PN $G_p$, embedding $G_v$ on $G_p$ means to find two one-to-one mappings: $M_v$ and $M_p$. Here $M_v$ is a one-to-one mapping from $N_v$ to a subset of $N_p$, For each virtual node $u \in N_v$ and the physical node $M_p(u)$ that it maps to: (i) the probability of violating the CPU demand (i.e., the available CPU of $M_p(u)$ is less than the CPU demand of $u$) is less than some small probability $\alpha$, which we term as violation probability; (ii) the distance of the location of $M_p(u)$ and the ideal location demanded by $u$ is less than a maximum predefined value, which is denoted by $R_u$. Here $M_l$ is from $L_p$ to a subset of all loop-free paths composed by the physical links in $L_p$. For each virtual link $lpq$ and each physical link in the physical path $M_l(lpq)$ that it maps to, the probability of violating bandwidth constraint should also be less than the violation probability $\alpha$. Take the embedding in Fig. 1(d) as an example. The node mapping solution is $\{a \rightarrow A, b \rightarrow C, c \rightarrow F\}$ and the link mapping solution is $\{ab \rightarrow ABC, ac \rightarrow AF, bc \rightarrow CBF\}$.  

2.2. Energy modeling

We extend the energy model with constant demands in the work [15,16] into our context with dynamic demands.
2.2.1. Node energy consumption modeling

For estimating the power consumption of the above two kinds of nodes, we apply the linear model reported in [21] as follows:

\[ P = \begin{cases} P_b + P_1 \cdot \text{CPU}, & \text{if the node is active} \\ 0, & \text{otherwise} \end{cases}. \tag{1} \]

where \( P_b \) denotes the baseline power and \( P_1 \) denotes the energy proportion factor for CPU. Note that, in Fig. 1(c), the physical nodes in active state are drawn in transparent while the nodes in inactive state are drawn in grey. We classify the physical nodes into two categories: (i) hosting nodes which are responsible for hosting virtual nodes with CPU demand in the node mapping and (ii) forwarding nodes which are acting as hidden hops [22] for forwarding packets in mapping a virtual link to a physical path. For example, in Fig. 1(c), C is a hosting node and B is a forwarding node. Let \( NH(t) \) and \( NF(t) \) denote the total number of hosting nodes and forwarding nodes at the time of \( t \), respectively. Note that for a physical node \( p \), it may be as a hosting node and forwarding node simultaneously. In order to avoid double counting, if a physical node is a hosting node, it will not be summed into \( NF(t) \) even if it is also a forwarding node. Let \( G_p(t) \) denote the number of VNs served by the PN at the time of \( t \). We calculate the node energy cost in time \( T \) as follows:

\[ \text{EN}(T) = \int_0^T \left[ P_b \cdot \text{NH}(t) + P_1 \sum_{i=1}^{G_p(t)} \sum_{u \in N_i^p} C_u(t) \right] dt \]

\[ \text{ES}(T) = \int_0^T E_s \cdot S(t) dt. \tag{3} \]

where \( S(t) \) denotes how many physical nodes switching from the inactive state into the active state at the time of \( t \).

In this paper, we consider the PN as a local network and the physical links may not span over a large geographical region, thus the link energy cost [16] is omitted in this model.

We notice that, in the production network, to guarantee the QoS of the applications, the InPs may not turn off the network equipment to save energy cost today. However, energy is a major cost for ISPs indeed. For example, in US, as one of the world’s largest providers of content delivery networking services, Akamai has to pay the electricity cost of about $10 Million each year [23]. In China, China Mobile Communications Corporation, the largest mobile service provider in the world, consumes over 13 TWH power consumption in 2011 [24]. In Italy, Telecom Italia consumes more than 2 TWh per year, which is equivalent to the energy consumed by 660,000 families in one year [25]. Another observation lies in that, in order to accommodate traffic shifts and to allow rerouting when links fail, the network capacity is usually overprovisioned. In real world production network, besides the production network traffic, some nodes are also used to carry a constant amount of background traffic [17]. For such nodes, we just keep them active and turn off the unused links for energy consumption. In such a way, it is natural to consider powering off routers or line cards for energy saving during periods of low utilization. Even though most of today’s routers cannot put line cards in “sleep mode”, or bring shut-down interfaces back up quickly, we believe that these advances will come in the years ahead, especially if they offer such a big energy savings [26].

2.3. The formulation

We next formulate the energy aware VN embedding with dynamic demands to an integer linear programming (ILP).

**Binary variables:**

\[ x_{\lambda i}^{\mu} \] equal to 1 if virtual node \( u \) of the \( i \)th VN is mapped to the physical node \( p \) and 0 otherwise.

\[ f_{pq}^{\mu \nu} = x_{\lambda i}^{\mu} \] equal to 1 if virtual link \( l_{\lambda i}^{\mu \nu} \) of the \( i \)th VN is routed on the physical link \( l_{pq} \) and 0 otherwise.

**Objective:**

\[ \text{Min} \quad E(T) = \text{EN}(T) + \text{ES}(T) \tag{4} \]

Before presenting the constraints, we have the following two lemmas:

**Lemma 1.** Given a set of independent variables \( X = (x_1, x_2, \ldots, x_n) \) in which \( x_i \sim N(\mu_i, \sigma_i^2) \), the sum of these independent variables \( \sum_{i=1}^{n} x_i \) also follows a Gaussian distribution with the mean as \( \sum_{i=1}^{n} \mu_i \) and the variance as \( \sum_{i=1}^{n} \sigma_i^2 \).

**Lemma 2.** Given a variable \( x \sim N(\mu, \sigma^2) \) and a violation probability \( \alpha \), \( \Pr(x > x^*) \leq \alpha \) holds if and only if \( \mu + \Phi^{-1}(1-\alpha)\sigma \leq x^* \), where \( \Phi^{-1} \) is the reverse function of the cumulative distribution function \( \Phi \) of the N(0, 1).

Based on Lemma 1 and 2, we have the following theorems:

**Theorem 1.** Given a sequence of VNs \( L = G_1', G_2', \ldots, G_n' \), \( \forall u \in N_i^p (i \in [1, n]) \), \( C_u \sim N(\mu_u, \sigma_u^2) \), if we map \( u \) to a physical node \( p \), the following capacity constraint must be satisfied:

\[ \forall p \in N_p : \quad x_{\lambda i}^{\mu} \cdot \left[ \sum_{i=1}^{G_p(t)} \sum_{u \in N_i^p} \mu_u + \beta \sum_{i=1}^{G_p(t)} \sum_{u \in N_i^p} \sigma_u^2 \right] \leq C_p \tag{5} \]

where \( \beta = \Phi^{-1}(1-\alpha) \).

Similarly, we also have a capacity constraint for the link:

**Theorem 2.** Given a sequence of VNs \( L = G_1', G_2', \ldots, G_n' \), \( \forall u \in N_i^p (i \in [1, n]) \), \( B_{pq} \sim N(\mu_{pq}, \sigma_{pq}^2) \), if we map \( l_{\lambda i}^{\mu \nu} \) to a physical path, each link \( l_{pq} \) of which must satisfy the following constraint:

\[ \forall l_{pq} \in L_p : \quad f_{pq}^{\mu \nu} \cdot \left[ \sum_{i=1}^{G_p(t)} \sum_{u \in N_i^p} \mu_{pq} + \beta \sum_{i=1}^{G_p(t)} \sum_{u \in N_i^p} \sigma_{pq}^2 \right] \leq B_{pq} \tag{6} \]

The location constraint of the virtual node is as follows:

\[ \forall i \in G_p(t), \forall u \in N_i^p, \forall p \in N_p : x_{\lambda i}^{\mu} \cdot \text{Dis}(D_p, D_u) \leq R_{iu}. \tag{7} \]
which denotes the Euclidean distance between these two nodes $D_p$ and $D_u$ cannot exceed the maximum distance $R_{uv}$.

The connectivity constraint for mapping virtual links is:

$$\forall i \leq G_v(t), \forall l_{uv} \in L_u^i, \forall p \in N_p :$$

$$\sum_{l_{uv} \in l_{uv}} f_{lu} - \sum_{l_{uv} \in l_{uv}} f_{pu} = \begin{cases} 1, & \text{if } x_{iu} = 1 \\ 0, & \text{otherwise} \end{cases}$$  \hspace{1cm} (8)

Finally, we list the variable constraints as follows:

$$\forall i \leq G_v(t), \forall p \in N_p : \sum_{u \in N_u} x_{pu} \leq 1$$  \hspace{1cm} (9)

$$\forall i \leq G_v(t), \forall u \in N_u^i, \forall l_{uv} \in L_p : f_{lu} = \begin{cases} 1, & \text{if } x_{pu} = 1 \\ 0, & \text{otherwise} \end{cases}$$  \hspace{1cm} (10)

These two constraints above ensure that a virtual node must correlate with just one substrate node.

$$\forall i \leq G_v(t), \forall u \in N_u^i, \forall p \in N_p : x_{pu} \leq 1$$

$$\forall i \leq G_v(t), \forall l_{uv} \in L_u^i, \forall p \in N_p : f_{lu} \leq 1$$

These two constraints above denote the binary domain constraints for the variables $x_{pu}$ and $f_{lu}$.

### 2.4. Performance metrics

For providing rental VN services, the InP will charge the SPs some fees, which is termed as the revenue achieved by the InP. In real world, the revenue modeling is a very complex task [27,28]. Similar to most of the previous studies [6,9,16], we apply a simple yet reasonable model as follows: For accommodating the ith VN, the revenue, denoted by $R_t(G_v)$, depends on how many nodes and links it demands, how many resources it requests and how long time it stays in the PN. That is, $R_t(G_v) = \sum_{u \in N_u} C_u(t) + \sum_{l \in L_p} B_{ul}(t)$. We then define the long-term average revenue as follows:

$$R(T) = \sum_{t=1}^{T} R(G_v)$$

where $T$ is the number of VN requests accepted by the PN successfully in time $T$.

Similarly, we define the long-term average energy cost as follows:

$$E(T)$$

$$\left\{ \begin{array}{ll} T & \text{if } \frac{C_{(t voyeur)} + C_{(t provider)}}{E_{(t voyeur)} + E_{(t provider)}} < 1 \\ \infty & \text{otherwise} \end{array} \right.$$

While $E(T)$ denotes the energy cost in time $T$ in Eq. (4).

### 3. Heuristic algorithm design

Solving ILP is NP-hard [29]. One branch is to find the optimal solution through standard exact algorithms. However, our preliminary results show that even for a very small input, e.g., mapping a VN with 4 nodes to a PN with 50 nodes, the GNU Linear Programming Kit (GLPK) [30], a standard ILP solver, takes several hours or even several days to compute the optimal solution. Even worse, they may incur exponential increasing running time for a large-sized network. Therefore, they are not applicable for online VN embedding. In this section, we design a simple yet efficient heuristic called EADVNE, which includes two steps. The first step handles node mapping and the second step handles link mapping.

#### 3.1. Node mapping step

In this step, we have two goals: guaranteeing high revenue and optimizing the energy consumption. Next we present how to achieve these two goals.

**Guaranteeing high revenue:** We have two approaches for this goal. One is to strike a resource balance between the multi-dimensional resource of the PN in case that for one physical node, one kind of resource (e.g., CPU) is run out while the other kind of resource (e.g., bandwidth) is still abundant; ii) the other one is to pre-consider and maximize the successful possibility of link mapping, since node mapping solutions seriously affect the quality of subsequent link mapping solutions.

As the first approach, in order to strike a balance between the CPU resource and bandwidth resource, we compute the angle between the demands of virtual node $u$ and the capacities of physical node $p$. Let the vector $\vec{u}_t = (u_{te}, u_{tp}, \ldots, u_{te})$ denote the resource demands of the virtual node $u$. Here $u_{te} = C_u(t)$, $B_{ut}(t)$ denotes that demands of $u$ at the time of $t$, where $C_u(t)$ denotes its CPU demand and $B_{ut}(t)$ denotes the sum of bandwidth demand between $u$ and all its neighbors, respectively. Let the vector $\vec{p}_t = (p_{te}, p_{tp}, \ldots, p_{te})$ denote the resource capacities of the physical node $p$. Here $p_{te} = C_p(t)$, $B_{pt}(t)$ denotes that capacity at the time of $t$, where $C_p(t)$ and $B_{pt}(t)$ denote its CPU capacity and bandwidth capacity respectively. We compute the angle between the vector $\vec{u}_t$ and $\vec{p}_t$ as follows: $\cos(\vec{u}_t, \vec{p}_t) = \frac{u_{te} \cdot p_{te} + u_{tp} \cdot p_{tp} + \ldots + u_{te} \cdot p_{te}}{|\vec{u}_t||\vec{p}_t|}$, where $|\vec{u}_t||\vec{p}_t|$ denotes the dot product of these two vectors, and $|\vec{u}_t|$ and $|\vec{p}_t|$ represent the magnitudes of these two vectors, respectively. Then we have

$$\cos(\vec{u}_t, \vec{p}_t) = \frac{\sum_{e=1}^{t} \cos(u_{te}, p_{te})}{t - e}.$$

When $\cos(\vec{u}_t, \vec{p}_t) = 1$, it means these two vectors are parallel. In this condition, the resources of the CPU and bandwidth of $p$ are used balanced and a high resource utilization will be achieved. We present an illustrative example in Fig. 2.

As the second approach, we take the core idea from [10] and normalize the magnitude of vectors for all physical nodes:

$$\text{mag}(\vec{p}) = \frac{|\vec{p}|}{\max_{q \in N_p}(|q|)}.$$  \hspace{1cm} (16)

We prefer to map a virtual node onto a physical node with the largest amount of resources to benefit the link mapping.

**Energy conservation:** Recall that, as shown in Eqs. (2) and (3), to minimize the energy cost, the key lies in exploiting the dynamic characteristic of the CPU demands to minimize the number of active hosting physical nodes, denoted by $NH(t)$. Specifically, we employ the Pearson correlation coefficient between the CPU demand of $u$ and the available CPU on $p$ as follows:

$$r(\vec{u}_t, \vec{p}_t) = \frac{n \sum C_u C_p - (\sum C_u)^2 \sum C_p}{\sqrt{n \sum C_u^2 - (\sum C_u)^2} \sqrt{n \sum C_p^2 - (\sum C_p)^2}}.$$  \hspace{1cm} (17)

Here, to simplify this equation, we use $C_u$, $C_p$, $\Sigma$ to represent $C_u(t)$, $C_p(t)$, $e_t - e_t$ and $\Sigma_{e_t}$, respectively. Note that, the value of $r(\vec{u}_t, \vec{p}_t)$ is between $-1$ and $1$. When $r(\vec{u}_t, \vec{p}_t) = -1$, the successful possibility of link mapping is high, and the energy cost is low.
it means these two vectors are negatively correlated and have the opposite variation trends. When \( \cos(\vec{u}, \vec{p}) = 1 \), it means these two vectors are positively correlated and have the same variation trend, which is preferred. We present an illustrative example in Fig. 3. Note that, this approach can also benefit the utilization of the PN besides energy conservation.

Finally, to achieve these two goals simultaneously, we rank the physical nodes by aggregating all of the \( \cos(\vec{u}, \vec{p}) \) and \( \text{mag}(\vec{p}) \) in the following one equation:

\[
\text{score}(\vec{p}) = \theta_1 \cos(\vec{u}, \vec{p}) + \theta_2 r(\vec{u}, \vec{p}) + \theta_3 \text{mag}(\vec{p}),
\]

(18)

where \( \theta_1, \theta_2, \theta_3 \in (0, 1) \) and \( \theta_1 + \theta_2 + \theta_3 = 1 \). We select the physical node with highest score value. The pseudocode of the above algorithm is in Algorithm 1.

### 3.2. Link mapping step

In this step, we also have two goals to achieve: guaranteeing revenue and energy conservation. For the first goal, we need to find the shortest path and thus save bandwidth resources. For the second goal, on one hand, we need to optimize the energy cost by minimizing \( \text{NF}(t) \) in Eq. (2); on the other hand, we need to exploit the dynamic characteristic of the bandwidth demand to optimize the resource utilization. In order to achieve these above goals, based on [31], we further design a weighted energy aware shortest path algorithm with the preference of active nodes and opposite variation trends between the bandwidth demand and the bandwidth load. Specifically, we first calculate the shortest paths between all pairs of physical nodes. Next, for a given physical node \( p_i \) and \( p_n \), we can find a path set by iteratively increasing the length of the paths until the length reaches a predefined threshold. In this path set, for each path, all physical links should satisfy the bandwidth demands within violation probability \( \alpha \). Then, for each path \( pa \), we calculate the \( \text{NF}(pa) \) which denotes how many physical nodes should be powered up from inactive state as forwarding nodes. Similar to Eq. (17), we also calculate the average Pearson correlation coefficient \( r(l_{\text{bw}}, pa) \) between the bandwidth demand and the bandwidth load. We rank each path \( pa \) as follows:

\[
\text{score}(pa) = \eta_1 \text{len}(pa) + \eta_2 \text{NF}(pa) + \eta_3 r(l_{\text{bw}}, pa).
\]

(19)
where $\eta_1, \eta_2, \eta_3 \in (0, 1)$ and $\eta_1 + \eta_2 + \eta_3 = 1$. We prefer to map a virtual link to the path with the lowest score. The pseudocode of the above algorithm is shown in Algorithm 2.

![Algorithm 2: Energy Aware Link Mapping (One by One)](image)

### 3.3. Time complexity analysis

Note that the time complexity of node mapping for one VN request is $O(|N_v| \cdot |N_p|)$ and the time complexity of link mapping is $O(|I_v| \cdot |N_p|^3)$ time. Thus, EAD-VNE is a polynomial-time algorithm.

### 4. Enhanced algorithm design with sliding windows

So far we have addressed the online VN embedding problem, which means as we process the VN requests as they arrive, in the way of “one by one”. For realistic applications, one may not always arrive definitely after another one. They may arrive nearly at the same time [9]. Another case is that they are delay-tolerant. In these scenarios, it becomes feasible that we queue a group of VN requests for a sliding time window, and map them together to further optimize the resource utilization and energy efficiency. In this subsection, motivated by such idea above, we propose another algorithm called EAD-VNE-G, which maps the VN requests more efficiently than EAD-VNE.

#### 4.1. Heuristic outline

The basic idea of our proposed algorithm is as follows. Assume there is a group of VN requests, each of which also has dynamic node demands and link demands, as described in Section 2.1. Inspired by the solutions of the classical bin packing problem\(^1\), to further leverage the dynamic characteristic of these demands, in the node mapping step, we propose to first accommodate these VN virtual nodes into super nodes, and then map such super nodes to the physical nodes; in the link mapping step, we still employed the algorithm in Section 3.2.

#### 4.2. Node mapping: map virtual nodes to super nodes

Let $L = (G_1, G_2, \ldots, G_n)$ denote a group of VN requests to be mapped, where $G_i = (N_i, L_i)$. Assume that, in $N_i$, it includes $m$ virtual nodes with dynamic CPU demands. Since different virtual nodes in the same VN request cannot be mapped to the same physical node, we must leverage an additional auxiliary vector $C = (c_1, c_2, \ldots, c_n)$, which is called the color vector. In this vector, different dimensions denote different colors. We first label all of the virtual nodes in the $i$th VN request with the color $c_i$, as shown in Fig. 4(a). Then, we enqueue these virtual nodes in the VN requests in $L$ according to the values of their CPU demands in non-decreasing order. Next, we enqueue them to a queue $Q_s$ as shown in Fig. 4(b). For each virtual node $u$ in $Q_s$, as shown in Fig. 4(c), dequeue it and try to accommodate it to an existing super node $s$ according to the following rules: (i) the sum of the CPU value of the super nodes must be less than a predefined threshold value, denoted by $\text{MAX\_SN}$; (ii) the Pearson correlation coefficient of $u$ and $s$ must be less than that of $u$ and other super nodes; (iii) in one super node, there cannot exist two virtual nodes in the same color; (iv) if we cannot find such super nodes, activate a new super node to accommodate $u$. We process each virtual node in such a way until the queue $Q_s$ is empty. The pseudocode of the above algorithm is shown in Algorithm 3.

![Algorithm 3: Enhanced Energy Aware Node Mapping (Group by Group)](image)

#### 4.3. Node mapping: map super nodes to physical nodes

So far, we dequeue the virtual nodes from $Q_s$ and generate super nodes. If we consider the super nodes here as the virtual nodes in Section 3.1, the next step is similar to Algorithm 1. We rank the super nodes according to the values of their CPU demands in non-decreasing order and enqueue them to another queue named $Q_p$. In this queue, we dequeue each super node $s$ and map it to one physical node as Algorithm 1 described.

The benefit of this algorithm is that it can accumulate several VN requests and process them at once to further leverage the dynamic characteristic and relationships among the VN demands to the greatest extent.
4.4. Link mapping with sliding windows

After aggregating a large amount of virtual nodes for different VN requests into several super nodes, these virtual links between virtual nodes for these different VN requests are still awaiting to be mapped to the physical network. For each virtual link, we need to find a physical path rather than a physical link, which is more difficult than the node mapping. If we process these virtual links together, similar to the node mapping as shown in Sections 4.2 and 4.3, it will incur a very high time complexity. For this reason, we still perform the link mapping in the simple yet efficient way as shown Algorithm 2 in Section 3.2.

4.5. Time complexity analysis

Since we need to accommodate the virtual nodes into super nodes, the algorithm EAD-VNE-G, which processes VN requests group by group, will consume slightly more running time than the algorithm EAD-VNE, which processes VN requests one by one. However, they share nearly the same level of time complexity.

5. Experimental results

This section evaluates our heuristic algorithms in terms of the performance metrics in Section 2.4, i.e., the long-term average revenue in Formula (13) and the long-term average energy cost in Formula (14). Note that, acceptance ratio is a very important metric of comparison of VN embedding. However, in the revenue modeling, it has considered the scales, duration, resource volumes and the acceptance ratio of VNs. Only the VN requests that has been successfully embedded can be computed into the revenue. Thus, the revenue can be regarded as an enhanced metric for the acceptance ratio and we omit the comparison in terms of acceptance ratio.

5.1. Experimental setup

We use the GT-ITM tool [32] to generate the topology of the PN and VNs. The detailed settings of single PN and VN, similar to studies [9,15,16,33], are described in Table 2. For characterizing the dynamics of the CPU and bandwidth demands, we generated these demands synthetically based on Gaussian distribution according to the observations in Section 2.1. We evaluate our algorithm by varying the scales of VN requests from a small size to a regular size. Small size means that the number of virtual nodes is uniformly distributed between 2 and 5, and regular size means that the number of virtual nodes is uniformly distributed between 2 and 10. Similar to the previous work [12], each physical node and virtual node are randomly distributed in a (100 × 100)
grid and \( R_{iu} \) in constraint (7) is set to 30. We assume VN requests arrive following a Poisson process with an average arrival rate of 4 VNs per time window (i.e., 100 time units or minutes), and each one has an exponentially distributed lifetime with an average of 500 time units. In EAD-VNE-G, we set \( \text{MAX}_\text{SN} \) (in Algorithm 3) to 30. We set the number of time windows \( w \) (in Algorithm 3) to 1, 2 and 3, respectively, to evaluate the impact of this parameter on the performance. In one instance, there are 50,000 time units, which respond to about 2000 VN requests. We ran 10 random different instances with these settings and calculated the mean of the 10 runs. We set the violation probability \( \alpha = 0.1 \) and correspondingly, \( \beta \) is about 1.28 in the Constraint (5) and (6). We set \( P_b \) and \( P_l \) defined in Eq. (1) to 165W and 15W/CPU unit, respectively, similar to [15,16]. Following previous work [34], we set \( E_s \) to one-hour energy consumption at the maximum load. We set all of \( \theta_1, \theta_2, \theta_3 \) in Eq. (18) and \( \eta_1, \eta_2, \eta_3 \) in Eq. (19) to 1/3.

We implemented our algorithm in C++ and performed side-by-side comparisons to the state-of-the-art VN embedding algorithm, e.g., EA-VNE proposed in [15,16]. EA-VNE considers the VN demands as constant values, neglecting the dynamic characteristic of CPU and bandwidth demands. Meta-heuristics [12,16] are orthogonal to our algorithms, which means we can also leverage such techniques to improve our algorithms in this paper, thus we do not compare our algorithms with them. Since studies [13,14] apply different models and thus their proposed algorithms do not work in our model, we exclude them from comparisons. In order to evaluate how far our solution is to the optimal solution, we also tried to compare EAD-VNE with GLPK [30], a standard ILP solver. However, through our preliminary simulations, GLPK takes several hours or even several days to compute the optimal solution even for a very small input, which is not applicable for online VN embedding, thus we omit this comparison. Since supporting multipath in link embedding is not the focus of this paper, we also omit the comparison with the multipath supported VN embedding algorithms.

5.2. Experimental results

We will first compare EAD-VNE to the state-of-the-art energy aware VN embedding problem in Section 5.2.1. Then we will further compare our enhanced algorithm EAD-VNE-G to EAD-VNE in Section 5.2.2.

5.2.1. Comparison between EAD-VNE and EA-VNE

Figs. 5 and 6 depict the evaluation results with 95% confidence interval between EAD-VNE and EA-VNE on small-sized and regular-sized VN inputs, respectively. We summarize the observations as follows.

**Energy comparison:** EAD-VNE reduces the energy cost significantly compared to EA-VNE. Fig. 5(a) shows that EAD-VNE significantly reduces the energy cost for the small-sized VNs. For example, at the time unit of 48,000, the energy cost of EAD-VNE is about 10% less than that of EA-VNE, the former being 146 and the latter 131. This is expected, because EAD-VNE leverages the dynamic characteristic of VN demands and makes a better consolidation into less number of physical nodes in the node and link mapping, which can be observed in Fig. 5(c). Through Fig. 6(a), we observe that with the VN scaling to the regular size, the space for optimizing the energy consumption also expands. Thus EAD-VNE achieves 16% less energy consumption than EA-VNE.

**Revenue comparison:** EAD-VNE obtains nearly the same revenue as EA-VNE. As shown in Figs. 5(b) and 6(b), we observe that EAD-VNE can produce similar revenue to EA-VNE. This is because in the node mapping, we strike a resource...
balance, leverage the Pearson correlation coefficient and pre-
consider the subsequent link mapping; in the link mapping
we also avoid mapping a virtual link to a long-path physical
path. These two strategies guarantee the high acceptance ra-
tio and low resource cost and thus achieve high revenues.

Running time comparison: Our algorithm consumes
similar running time than EA-VNE. We conducted the sim-
ulation on the same server with Intel 3GHz dual-core CPU,
2GB memory, 160GB disk, and Linux 2.6 OS. Table 3 depicts
the average running time of VN embedding algorithms. We
show that EAD-VNE consumes a little more time than EA-
VNE for computing the angle between $\vec{u}$ and $\vec{p}$, and Pearson
 correlation coefficient, which is deemed as reasonable and
acceptable.

5.2.2. Comparison between EAD-VNE-G and EAD-VNE

So far, we evaluated the heuristic algorithm EAD-VNE by
comparing it to EA-VNE. Next, we will proceed to evaluate
the algorithm with sliding windows, i.e., EAD-VNE-G. We re-
port the experimental results (with 95% confidence interval)
in terms of energy consumption, revenue and the number of
active physical nodes in Fig. 7(a)–(c), respectively. As shown
in Fig. 7, we show that algorithm EAD-VNE-G reduces 12% en-
ergy consumption than EA-VNE, and 25% energy consump-
tion than EA-VNE for regular-sized VN requests. The reason
lies in that EAD-VNE-G can further exploit the dynamic char-
acteristics among VN requests to consolidate the VN requests
into smaller number of virtual nodes, as shown in Fig. 7(c).
Thus, it saves more energy consumption.

As shown in Table 3, EAD-VNE-G also consumes slightly
more running time than EA-VNE for generating those vir-
tual nodes for VN requests into several less super nodes.

5.2.3. The impact of the numbers of sliding windows $w$ on the
performance

To evaluate the impact of the numbers of sliding windows
$w$ in Algorithm 3, we vary it from 1 to 3. The experiment
results are also shown in Fig. 7(a). As shown in this figure, we
observe two phenomena. The first is, with longer sliding win-
dow, the more energy consumption EAD-VNE-G will achieve.
This is expected, because if it has a longer sliding window, the
algorithm will have a larger optimization space. The extreme
case is we know all the VN requests in advance, it will be-
come the offline VN embedding problem. Similar to the bin
packing problem, the result of offline case is usually better
than the online case. However, the second phenomenon is
that, when $w$ increases, the relative advantage with a larger
$w$ over the algorithm with a smaller $w$ drops. This is also
expected, the reason lies in that when $w$ increases, the rel-
ative optimization space also shrinks. Moreover, every coin
has two sides. The disadvantage of those algorithms based
on sliding windows is also obvious: it will incur a very large
delay for the VN requests. However, it offers a flexible knob
for the InPs to trade off the energy consumption and the de-
lay of the VN requests after all.

6. Related work

As a major challenge in network virtualization, VN em-
bedding has received significant attention in the research
community and industry. Due to the NP-hardness of this
problem, early studies aimed to improve the scalability [7]
or load balance of the physical network [4] of this problem
while making some reasonable assumptions. Since Yu et al.
[6] began to rethink and define a general framework for this
problem, the primary goal of most of the following studies
became to find efficient methods to maximize the revenue
generated for the InP. These methods include leveraging the
techniques of path splitting and migration [6], subgraph
isomorphism detection [8], deterministic and randomized
rounding technique [9], topology-aware node ranking [10],
global resource capacity based heuristic [11] and other
meta-heuristics [12,35].

However, none of these studies listed above has consid-
ered the energy-related cost in carrying out VN embedding.
Su et al. [15,16] took the first addressing the energy conser-
vation issue in the VN embedding context in the literature. In
parallel with these two studies, Botero et al. [36] applied the
exact algorithm to solve the energy efficient VN embedding
algorithm. However, it may incur high exponential running
time when the network scale increases. Like most of prior
art, they still considered the CPU and bandwidth demands as
constant values, however, in real-world applications, the de-
mands usually make great dynamic changes over time. Thus,
in this paper, we bridge this gap to study the problem of energy-aware VN embedding with dynamic demands. Recently, Botero et al. [31] proposed a novel energy-aware reconfiguration approach and leveraged load balancing to produce energy-efficient embeddings. Also, a power-aware shortest path technique was proposed for virtual link mapping. This work is orthogonal to our work, which means that, by leveraging the reconfiguration approach, the energy efficiency of our algorithm can be further improved.

Recently, there are also some studies considering the dynamic characteristic in VN embedding. Zhang et al. [13] modeled the VN requirement as the combination of a basic subrequirement and a variable sub-requirement. Based on such model, they proposed an opportunistic resource sharing-based mapping framework and designed two first-fit based algorithms for maximizing the utilization of the physical network. However, in this paper, we use a more practical Gaussian model and leverage the dynamic daily diurnal pattern. Moreover, our focus is on the energy issue in VN embedding. Sun et al. [14] modeled the bandwidth demand as a Gaussian distribution and designed a sliding window approach for maximizing the total revenue. However, they did not consider the dynamic characteristic of the CPU demand and the energy consumption.

There are also a few studies on inter-domain VN embedding [33,37,38], where the physical networks are distributed in geographical locations. For future work, we may extend our problem into the inter-domain VN embedding.

For more information about network virtualization and VN embedding problem, we highly recommend readers to [2,39] for reference.

7. Conclusion

In this paper, we study the problem of energy aware VN embedding with dynamic demands. Specifically, we first present the network model in which both of the node and link requirements follow the Gaussian distribution and daily diurnal pattern. We then design two energy-aware heuristic VN embedding algorithms by exploiting and leveraging the dynamic characteristic. One algorithm processes the VN requests one by one while the other one processes them group by group. Through extensive simulations, we show that our algorithm can save up to 25% energy consumption for InPs than the existing algorithm while generating nearly the same revenue. For future work, we plan to investigate the effects by tuning different parameters and extend this problem into inter-domain VN embedding context.

Acknowledgments

This work was supported in part by the following funding agencies of China: National Natural Science Foundation under grant 6170274, National Basic Research Program (973 Program) under grant 2011CB302506, and the Fundamental Research Funds for the Central Universities (2015RC21).

References


[36] Zhongbao Zhang received the Ph.D. Degree in Computer Science from Beijing University of Posts and Telecommunications, China, in 2014. He is an assistant professor in Beijing University of Posts and Telecommunications now. His major is Computer Science. His research interests include network virtualization, social network and big data.

[37] 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 at the Beijing University of Posts and Telecommunications. His research interests include cloud computing and big data.

[38] Junchi Zhang received the Bachelor Degree in Electronic Science and Technology from Beijing University of Posts and Telecommunications, China, in 2014. He is currently planning to go abroad for further development. His research interests include network virtualization, and social network.

[39] Kai Shuang received the Ph.D. Degree in Computer Science from Beijing University of Posts and Telecommunications, China, in 2006. He is currently an Associate Professor at the Beijing University of Posts and Telecommunications. His research interests include cloud computing and mobile Internet.

[40] Peng Xu received the Ph.D. Degree in Computer Science from Beijing University of Posts and Telecommunications, China, in 2004. He is currently an Associate Professor at the Beijing University of Posts and Telecommunications. His research interests include cloud computing.