Embedding Virtual Software-Defined Networks over Distributed Hypervisors for vDC Formulation

Huibai Huang, Shengru Li, Kai Han, Quanying Sun, Daoyun Hu, Zuqing Zhu†
School of Information Science and Technology, University of Science and Technology of China, Hefei, China
†Email: {zqzhu}@ieee.org

***Abstract*—We study how to embed virtual software-defined networks (vSDNs) cost-effectively over a substrate network with distributed network virtualization hypervisors (NVHs) for virtual datacenter (vDC) formulation to support Big Data applications. Specifically, we try to jointly optimize the embedding schemes of the control and data planes of each vSDN, *i.e.*, to minimize the data plane's resource consumption and limit the number of NVHs used in the control plane simultaneously. We first formulate an integer linear programming (ILP) model to solve the problem exactly, and then design a heuristic to reduce the time complexity. Simulation results suggest that our proposed algorithms can embed vSDNs cost-effectively and significantly outperform the existing scheme based on global resource capacity (GRC).**

***Index Terms*—Virtual Network Embedding (VNE), Distributed Hypervisors, Virtual Software-Defined Networks (vSDNs).**

I. Introduction

It is known that Big Data not only means tremendous volumes of data, but also implies the huge capacity for capturing, storing and analyzing data, which can hardly be provided by conventional data systems [1]. Hence, large enterprises such as Google and Amazon have built geographically-distributed datacenter (DC) systems to run their Big Data applications and provide low-latency, high-quality and non-disruptive services to end users. However, the cost of building and operating physical DC systems would be prohibitively high for small service providers. Moreover, due to the dynamics of Big Data applications, a short time-to-market, elastic and cost-effective solution is desirable. Fortunately, network virtualization allows multiple logically-isolated virtual networks to coexist on shared substrate infrastructure and provides infrastructure providers a powerful way to lease network resources dynamically to service providers (*i.e.*, infrastructure-as-a-service (IaaS)) for Big Data applications [2–4].

With network virtualization, the infrastructure provider can slice a geographically-distributed DC system into multiple virtual DC (vDC) systems dynamically according to the service providers' requirements and lease them on demand [5]. Note that, there are normally two types of resource virtualization in vDC formulation, *i.e.*, the IT resource virtualization in the DCs to create vDCs and the virtual network embedding (VNE) in the substrate inter-DC network to interconnect vDCs. Meanwhile, to enable effective network orchestration in each vDC system, one might expect the virtual network that interconnects its vDCs to be a software-defined network (SDN) [6]. It is known that SDNs leverage the centralized control plane (CP) to enhance network programmability [7]. Hence, in the vDC system, the CP of the virtual SDN (vSDN) can collaborate with the vDC controllers to make effective network orchestration possible for the vDC operator. This is especially important when the vDC operator needs to run Big Data applications that would result in highly-dynamic bandwidth utilization and bursty IT resource consumption.

Hence, it would be not only interesting but also necessary to study how to embed vSDNs cost-effectively for vDC formulation. Note that, although the problem of embedding common virtual networks into a substrate network, *i.e.*, the well-known VNE problem, has already been investigated intensively in literature [8–10], these studies only considered how to embed the data planes (DPs) of virtual networks. While to provision a vSDN, we have to embed its CP as well [11]. Moreover, in practical cases, we might need to consider the quality-of-service (QoS) related to control latency [12], and thus more than one virtual controllers (vCs) would be instantiated for the vSDN. Hence, to embed a vSDN, we need to address the VNE of its DP and CP in a correlated manner, which is apparently more sophisticated than the traditional VNE studied in [8–10].

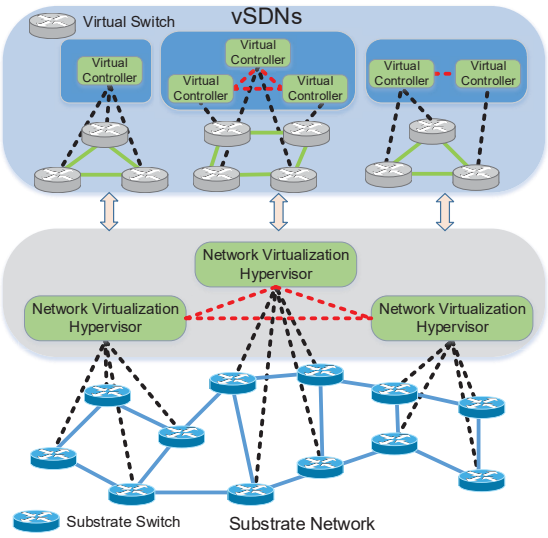

Fig. 1. Architecture of distributed NVH system for vSDNs.

Fig. 1 shows the network system that we consider in this work to address the embedding of vSDNs for vDC formulation. Basically, the VNE of the DP of a vSDN still consists of node mapping and link mapping. However, the resource considered in the node mapping will be the ternary

content-addressable memory (TCAM) space for storing flow rules. This is because TCAM is usually expensive and limited hardware resource in substrate SDN switches, while the DP of the vSDN needs to consume certain amount of TCAM on the substrate switches along the paths that carry its virtual links [13]. The VNE of the CP of a vSDN will be performed based on the locations of network virtualization hypervisors (NVHs) in the substrate network. Note that, NVHs slice the flow rule space of each substrate switch into logic space regions, and bridge the control communications between substrate switches and vCs of the vSDNs with control message translation [14]. Hence, it would be reasonable to assume that each vC of a vSDN is connected to an NVH locally. Meanwhile, since the QoS related to control latency is important for SDN operations [15], we should consider distributed NVHs in the substrate network. Note that, some of the latest SDN platforms, *e.g.*, Open Network Operating System (ONOS) [15], have already supported "physically-distributed-but-logically-centralized" CP architecture. Although distributed NVHs can bring advantages in terms of availability and scalability, using many NVHs in the CP of a vSDN can cause unnecessary east-/west-bound communication overheads.

In this paper, we study the embedding of vSDNs over a substrate network with distributed NVHs. Specifically, we try to jointly optimize the VNE schemes of the CP and DP of each vSDN, *i.e.*, to minimize the DP's resource consumption and limit the number of NVHs used in the CP simultaneously. We first formulate an integer linear programming (ILP) model to solve the problem exactly. Then, a heuristic algorithm is designed for reducing the time complexity. Simulation results show that our proposed algorithms can embed vSDNs cost-effectively and significantly outperforms the existing scheme based on global resource capacity (GRC).

The rest paper is organized as follows. Section II provides a brief survey on the related work. We describe the network model and formulate the ILP model in Section III. The heuristic algorithm is designed in Section IV and Section V discusses the simulations for performance evaluation. Finally, Section VI summarizes this paper.

II. Related Work

Although it is known that SDN and network virtualization are independent of each other, recent research has suggested that the symbiosis of them would be beneficial in terms of network programmability, scalability and adaptivity [14]. Therefore, the system implementations of several latest platforms, such as Nicira's Network Virtualization Platform [16] and OpenVirteX [17], have already moved towards this direction. Note that, the embedding of vSDNs is intrinsically different from the traditional VNE problems studied in [8, 9, 18], since to embed a vSDN, we need to address the VNE of its DP and CP in a correlated manner. In [13], Demirci *et al.* investigated the embedding of vSDNs under the assumption that only one centralized NVH would be used in the substrate network and thus each vSDN only uses a single vC. Nevertheless, this assumption becomes impractical when the substrate network covers a relatively large geographical area (*e.g.*, a geographically-distributed DC system). To the best of our knowledge, the embedding of vSDNs over a substrate network with distributed NVHs has not been studied yet.

III. Problem Description

A. Network Model

The architecture of the network system for vSDN embedding is shown in Fig. 1. Here, we model the substrate network as an undirected graph, denoted as $G_s(V_s, E_s)$, where V_s and E_s are the sets of substrate nodes (SNs) and links (SLs), respectively. The TCAM capacity of each SN $v_s \in V_s$ is T_{v_s} and the bandwidth capacity of each SL is $B_{(u_s,v_s)}$ where $(u_s, v_s) \in E_s$. There is a set of V_s^h distributed NVHs in the substrate network, where each $v_s \in V_s^h$ represents the physical location of an NVH (*i.e.*, $V_s^h \subset V_s$). We also assume that for each NVH on $v_s \in V_s^h$, it can control a set of SNs, denoted as Φ_{v_s}. Fig. 2(a) shows the example of a substrate network that includes two NVHs located on *Nodes* 2 and 7, respectively, and we assume $\Phi_2 = \{1, 2, 3, 4\}$ and $\Phi_7 = \{5, 6, 7, 8\}$.

For each vSDN that needs to be embedded in the network system, it is also modeled as an undirected graph $G_r(V_r, E_r)$. Similarly, each virtual link (VL) $(u_r, v_r) \in E_r$ of the vSDN has a bandwidth demand of $b_{(u_r,v_r)}$. The TCAM requirement of a virtual node (VN) $v_r \in V_r$ is t_{v_r}. Note that, the vSDN consumes TCAM not only on the SNs that its VNs are embedded on, but also on the SNs that are the intermediate nodes on the substrate paths to carry its VLs. Hence, for the intermediate SNs involved in the link mapping of VL (u_r, v_r), we assume that they consume a TCAM capacity of $\widetilde{t}_{(u_r,v_r)}$, which can be approximated as

$$\widetilde{t}_{(u_r,v_r)} = (t_{u_r} + t_{v_r}) \cdot \theta, \tag{1}$$

where θ depends on the NVH implementation, and normally, we have $\theta \in (0, 0.5]$. Fig. 2(b) illustrates two vSDNs and for simplicity, we omit the bandwidth and TCAM requirements.

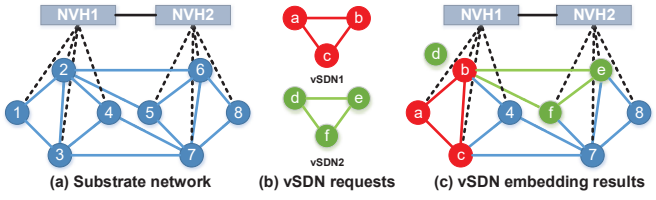

Fig. 2. Example on embedding vSDNs over a substrate network with distributed NVHs.

Fig. 2(c) shows the results for embedding the vSDNs in Fig. 2(b) onto the substrate network in Fig. 2(a). It can be seen that *vSDN* 1 is embedded on the SNs that can be controlled by the NVH on *Node* 2, while the SNs used to embed *vSDN* 2 have to be controlled by two NVHs. Hence, to operate *vSDN* 2, the two NVHs need to collaborate with each other, which would cause additional east-/west-bound communication overheads. Consequently, we should limit the number of NVHs used in the CP of each vSDN to avoid unnecessary overheads.

B. Problem Formulation in ILP

Notations:

- $G_s(V_s, E_s)$: topology of the substrate network.
- V_s^h: set of NVH locations in the substrate network.
- N_{v_s}: set of SNs that are adjacent to $v_s \in V_s$.
- Φ_{v_s}: set of SNs that can be controlled by an NVH on v_s.
- T_{v_s}: TCAM capacity of SN $v_s \in V_s$.
- $B_{(u_s,v_s)}$: bandwidth capacity of SL $(u_s, v_s) \in E_s$.
- $G_r(V_r, E_r)$: topology of the DP of a vSDN request.
- $b_{(u_r,v_r)}$: bandwidth demand of VL $(u_r, v_r) \in E_r$.
- t_{v_r}: TCAM demand of VN $v_r \in V_r$.
- $\widetilde{t}_{(u_r,v_r)}$: TCAM demand of the intermediate SNs on the substrate path carrying VL (u_r, v_r).
- $\hat{H}$: maximum number of NVHs used for embedding the CP of a vSDN request.

Variables:

- $\delta_{v_s}^{v_r}$: boolean variable that equals 1 if VN v_r is embedded on SN v_s, and 0 otherwise.
- $\rho_{(u_s,v_s)}^{(u_r,v_r)}$: boolean variable that equals to 1 if the embedding of VL (u_r, v_r) uses SL (u_s, v_s), and 0 otherwise.
- $\omega_{v_s}^{(u_r,v_r)}$: boolean variable that equals 1 if SN v_s is an intermediate node of the substrate path that VL (u_r, v_r) is mapped onto, and 0 otherwise.
- π_{v_s}: boolean variable that equals 1 if the NVH on $v_s \in V_s^h$ is used by the vSDN, and 0 otherwise.

Objective:

Note that, the vSDN $G_r(V_r, E_r)$ consumes two types of resources in the substrate network, *i.e.*, the TCAM on SNs and the bandwidth on SLs. The normalized TCAM utilization can be obtained as

$$\mu_t = \frac{\sum\limits_{v_s \in V_s} \sum\limits_{(u_r,v_r) \in E_r} \omega_{v_s}^{(u_r,v_r)} \cdot \widetilde{t}_{(u_r,v_r)}}{\sum\limits_{v_s \in V_s} T_{v_s}}. \tag{2}$$

Here, we ignore the TCAM usages on the SNs that have VNs embedded on, because their summation is a constant value as long as all the VNs have been successfully embedded. While the normalized bandwidth utilization is

$$\mu_b = \frac{\sum\limits_{(u_s,v_s) \in E_s} \sum\limits_{(u_r,v_r) \in E_r} \rho_{(u_s,v_s)}^{(u_r,v_r)} \cdot b_{(u_r,v_r)}}{\sum\limits_{(u_s,v_s) \in E_s} B_{(u_s,v_s)}}. \tag{3}$$

Hence, the optimization objective is to minimize total resource cost, which is

$$Minimize \quad \mu = \alpha_1 \cdot \mu_t + \beta_1 \cdot \mu_b, \tag{4}$$

where α_1 and β_1 are the unit price of TCAM and bandwidth resources, respectively.

Constraints:

1) *Node Mapping Constraints:*

$$\sum_{v_s \in V_s} \delta_{v_s}^{v_r} = 1, \quad \forall v_r \in V_r. \tag{5}$$

Eq. (5) ensures each VN in the DP of the vSDN is mapped onto an SN.

$$\sum_{v_r \in V_r} \delta_{v_s}^{v_r} \leq 1, \quad \forall v_s \in V_s. \tag{6}$$

Eq. (6) ensures different VNs in the DP of the vSDN is mapped onto different SNs in the substrate network.

2) *Resource Constraints:*

$$\sum_{(u_r,v_r) \in E_r} \omega_{v_s}^{(u_r,v_r)} \cdot \widetilde{t}_{(u_r,v_r)} + \sum_{v_r \in V_r} \delta_{v_s}^{v_r} \cdot t_{v_r} \leq T_{v_s}, \ \forall v_s \in V_s. \tag{7}$$

Eq. (7) ensures that the TCAM usage on each SN for the vSDN would not exceed its TCAM capacity.

$$\sum_{(u_r,v_r) \in E_r} \rho_{(u_s,v_s)}^{(u_r,v_r)} \cdot b_{(u_r,v_r)} \leq B_{(u_s,v_s)}, \quad \forall (u_s, v_s) \in E_s. \tag{8}$$

Eq. (8) ensures that the bandwidth usage on each SL for the vSDN would not exceed its bandwidth capacity.

3) *Flow Conservation Constraints:*

$$\sum_{v_s \in N_{u_s}} \rho_{(u_s,v_s)}^{(u_r,v_r)} - \sum_{v_s \in N_{u_s}} \rho_{(v_s,u_s)}^{(u_r,v_r)} = \delta_{u_s}^{u_r} - \delta_{u_s}^{v_r}, \quad \forall u_s \in V_s, \ \forall (u_r, v_r) \in E_r. \tag{9}$$

Eq. (9) ensures that one VL is mapped onto one and only one substrate path, *i.e.*, we only consider single-path mapping.

$$\rho_{(u_s,v_s)}^{(u_r,v_r)} = \rho_{(v_s,u_s)}^{(u_r,v_r)}, \quad \forall (u_s, v_s) \in E_s, \ \forall (u_r, v_r) \in E_r. \tag{10}$$

Eq. (10) ensures that the in- and out-flows of an SN for carrying the same VL traverse only one substrate path.

4) *Intermediate Node Constraints:*

$$\sum_{v_s \in N_{u_s}} \rho_{(u_s,v_s)}^{(u_r,v_r)} - \omega_{u_s}^{(u_r,v_r)} = \delta_{u_s}^{u_r}, \ \forall u_s \in V_s, \forall (u_r, v_r) \in E_r. \tag{11}$$

Eq. (11) ensures that the intermediate nodes of the substrate path that carries VL (u_r, v_r) is marked correctly.

5) *NVH-related Constraints:*

$$\sum_{v_s \in \Phi_{u_s}} \sum_{u_r \in V_r} \delta_{v_s}^{u_r} \leq \pi_{u_s} \cdot |\Phi_{u_s}|, \quad \forall u_s \in V_s^h. \tag{12}$$

$$\sum_{v_s \in \Phi_{u_s}} \sum_{(u_r,v_r) \in E_r} \omega_{v_s}^{(u_r,v_r)} \leq \pi_{u_s} \cdot |\Phi_{u_s}| \cdot |E_r|, \quad \forall u_s \in V_s^h. \tag{13}$$

$$\sum_{v_s \in \Phi_{u_s}} \sum_{u_r \in V_r} \delta_{v_s}^{u_r} + \sum_{v_s \in \Phi_{u_s}} \sum_{(u_r,v_r) \in E_r} \omega_{v_s}^{(u_r,v_r)} \geq \pi_{u_s}, \quad \forall u_s \in V_s^h. \tag{14}$$

Eqs. (12)-(14) ensure that the NVH on SN u_s would be selected for the CP of the vSDN, if any of the VNs in V_r has been mapped onto any SN in set Φ_{u_s} or any of the VLs in E_r has been mapped onto the substrate path that includes the SNs in set Φ_{u_s} as intermediate nodes.

$$\sum_{v_s \in V_s^h} \pi_{v_s} \leq \hat{H}. \tag{15}$$

Eq. (15) enures that the number of NVHs used for embedding the CP of the vSDN would not exceed the preset upper-limit to restrict the east-/west-bound communication overheads.

IV. Heuristic Algorithm

A. Algorithm Design

Although the ILP model above can solve the problem of vSDN embedding exactly, its time complexity increases exponentially with the problem scale. Hence, in this section, we design a time-efficient heuristic algorithm. For each vSDN, as we need to address the VNE of its DP and CP in a correlated manner, we should first carefully consider the locations of distributed NVHs to embed the CP and then embed the DP within the subnet that can be controlled by the selected NVH(s) to utilize substrate resources efficiently. Therefore, we propose a topology abstraction scheme to preprocess the substrate network for vSDN embedding. Specifically, the topology abstraction aggregates the subnet that can be controlled by an NVH into a super node and then restructures the substrate topology accordingly. Fig. 3 shows an example of the topology abstraction. There are three distributed NVHs in the substrate network and their control regions are marked with different colors. The topology abstraction aggregates the subnets defined by the control regions into three super nodes, as illustrated in Fig. 3(b). Note that, we use an aggregated link to replace all the SLs that are between two subnets in the original substrate topology.

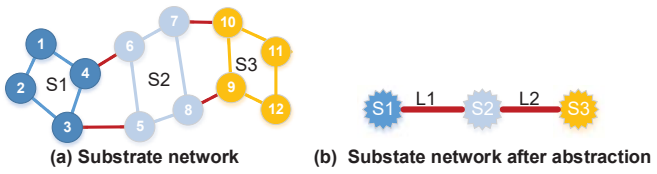

Fig. 3. Example on topology abstraction for preprocessing.

Then, with the abstracted topology G'_s, we embed the CP of the vSDN based on the following metrics. For a subnet defined by $\Phi_{v_s}, v_s \in V_s^h$, the ratio of the TCAM capacity in the subnet to the total TCAM capacity is

$$\gamma_{v_s} = \frac{\sum_{u_s \in \Phi_{v_s}} T_{u_s}}{\sum_{u_s \in V_s} T_{u_s}}, \tag{16}$$

Similarly, the bandwidth capacity ratio is

$$\eta_{v_s} = \frac{\sum_{(u_s,u'_s) \in E(\Phi_{v_s})} B_{(u_s,u'_s)}}{\sum_{(u_s,u'_s) \in E_s} B_{(u_s,u'_s)}}, \tag{17}$$

where $E(\Phi_{v_s})$ denotes the set of SLs that are within the subnet defined by Φ_{v_s}. Note that, the embedding of the CP is essentially to find proper subnet(s) in G_s for the vSDN. Hence, with the consideration of the subsequent DP embedding, we should try to choose the subnet(s) that have relatively large TCAM and bandwidth capacities. For this purpose, we take the geometric average of γ_{v_s} and η_{v_s} as the virtual capacity of the super node for Φ_{v_s} in G'_s, which can be used to quantify the super node's embedding potential.

$$\mathcal{T}_{v_s} = \sqrt{\gamma_\Phi \cdot \eta_\Phi}. \tag{18}$$

For two adjacent super nodes in G'_s, the aggregated link between them carries the bandwidth capacity of all the SLs between the two corresponding subnets in G_s. Therefore, with a similar approach as that is illustrated in Eq. (17), we can calculate the bandwidth capacity ratio of the aggregated link and denote it as $\mathcal{B}_{(v_s,u_s)}$, where $v_s, u_s \in V_s^h$ represents the locations of the NVHs in two adjacent subnets.

According to [9], an SN with larger global resource capacity (GRC) has a higher embedding potential for a VN and its adjacent SLs also have higher embedding potentials for the VLs that are adjacent to the VN. Hence, if we treat $\mathcal{T}_{v_s}$ and $\mathcal{B}_{(v_s,u_s)}$ as the node and link capacities in G'_s, respectively, we can calculate the GRC of each super node in G'_s. *Algorithm* 1 shows the overall procedures of the proposed heuristic algorithm. *Line* 1 is the initialization to obtain the abstracted topology G'_s. Then, the for-loop that covers *Lines* 2-28 embeds the pending vSDN requests one by one. In each iteration, *Lines* 3-5 are for the initialization to get the GRC of each super node in G'_s based on the current network status and to set three super node sets, *i.e.*, $\hat{V}_1$, $\hat{V}_2$ and $\hat{V}_3$, for subsequent processing. Next, the while-loop covering *Lines* 6-27 tries to embed the vSDN based on the three super node sets. Specifically, we try to embed the DP of the vSDN with the subnet(s) in G_s, which include the SNs and SLs that are covered by the super nodes in $\hat{V}_2$ and the super node with the largest GRC in $\hat{V}_3$, as shown in *Lines* 8-17. Note that, the actual embedding scheme for the DP of the vSDN is in *Algorithm* 2. If this cannot be done, *Lines* 21-22 would expand $\hat{V}_2$ by adding the super node in $\hat{V}_3$ with the largest GRC to $\hat{V}_2$. Then, we update $\hat{V}_3$ by adding the super nodes that are in $\hat{V}_1$ and adjacent to any nodes in $\hat{V}_2$ to $\hat{V}_3$ (*i.e.*, in *Lines* 24-26).

In *Algorithm* 2, we first embed the VN that has the largest GRC and then map the adjacent VNs in sequence to "grow" the DP of the vSDN out. Note that, we should not use the GRC-VNE algorithm in [9] here, because it might result in mapping VLs to substrate paths that are unnecessarily long. Since the vSDN consumes TCAM not only on the SNs that its VNs are embedded on but also on the SNs that are the intermediate nodes on the substrate paths to carry its VLs, mapping a VL to a long substrate path would waste TCAM resources. Hence, in *Line* 8 in *Algorithm* 2, we map v_r onto a feasible SN in V''_s with the largest metric calculated as

$$\xi = \alpha_2 \cdot \text{GRC} + \beta_2 \cdot \min(\text{hop-count}), \tag{19}$$

where α_2 and β_2 are the weights, "GRC" denotes the GRC of the SN, and "hop-count" represents the hop-count of the shortest substrate path between the SN and an SN that has already been used to carry a VN in $G_r(V_r, E_r)$. Since the

complexity of calculating GRC is proven to be $O(|V|^2 \cdot \log \frac{1}{\sigma})$ [9], where $\sigma \ll 1$ is the preset accuracy threshold, the time complexity of *Algorithm* 1 to embed a vSDN can be obtained as $O((|V_s|+|V_r|)^2 \cdot \log \frac{1}{\sigma} + |V_s|^3 \cdot |V_r| + |E_s| \cdot |E_r| \cdot \log(|V_s|))$.

Algorithm 1: Heuristic for Embedding vSDNs

apply topology abstraction on G_s to get $G'_s(V'_s, E'_s)$;
for *each pending vSDN request* $G_r(V_r, E_r)$ **do**
  calculate the capacities of each super node and link in G'_s to get $\{\mathcal{T}_{v_s}\}$ and $\{\mathcal{B}_{(v_s,u_s)}\}$;
  calculate the GRC of each super node $v'_s \in V'_s$;
  $\hat{V}_1 = V'_s$, $\hat{V}_2 = \emptyset$, $\hat{V}_3 = V'_s$;
  while $|\hat{V}_2| < \hat{H}$ **do**
    sort nodes in $\hat{V}_3$ in descending order of GRC;
    for *each super node* $\hat{v}_s \in \hat{V}_3$ **do**
      restore the subnet(s) in G_s with the SNs and SLs that are covered by the super nodes in $\hat{V}_2$ and $\hat{v}_s$;
      represent the topology of the restored subnets as $G''_s(V''_s, E''_s)$;
      try to embed the DP of vSDN G_r in G''_s with *Algorithm* 2;
      if *the DP of vSDN* $G_r(V_r, E_r)$ *can be embedded successfully* **then**
        update the TCAM and bandwidth capacities in G_s;
        $\hat{V}_1 = \emptyset$;
        break;
      end
    end
    if $\hat{V}_1 = \emptyset$ **then**
      break;
    else
      add the top-ranked super node in $\hat{V}_3$ in $\hat{V}_2$;
      $\hat{V}_3 = \emptyset$;
    end
    for *each super node* $\hat{v}_s \in \hat{V}_2$ **do**
      add the super nodes in $\hat{V}_1$ that are adjacent to $\hat{v}_s$ but not in $\hat{V}_2$ in $\hat{V}_3$;
    end
  end
end

V. Performance Evaluation

We perform numerical simulations to evaluate the algorithms's performance. Considering its time complexity, we first evaluate the performance of the ILP with a small-scale substrate network. The substrate network consists of 3 NVHs, each of which controls a subnet that includes 8 SNs in a topology that is randomly generated by the GT-ITM tool [19], with an SN connectivity ratio of 0.35. Every two subnets are connected with 5 random SLs. The TCAM capacity of each SN and the bandwidth capacity of each SL are randomly set

Algorithm 2: Procedure for Embedding DP of a vSDN

$\hat{V}_{r,1} = V_r$, $\hat{V}_{r,2} = \emptyset$, $f = 1$;
get GRC value of each SN in G''_s with the TCAM and bandwidth capacities of the SNs and SLs in G''_s;
sort SNs in V''_s in descending order of their GRCs;
get GRC value of each VN in G_r with the TCAM and bandwidth demands of the VNs and VLs in G_r;
select the VN in V_r with largest GRC;
move the selected VN v_r from $\hat{V}_{r,1}$ to $\hat{V}_{r,2}$;
while $\hat{V}_{r,1} \neq \emptyset$ *AND* $f = 1$ **do**
  try to map the VN v_r that is newly added in $\hat{V}_{r,2}$ onto a feasible SN in V''_s with the largest metric calculated by Eq. (19);
  try to embed the VL(s) ending at v_r with the shortest substrate path(s);
  if *node and link mappings related to* v_r *are successful* **then**
    select the VN in V_r that is adjacent to a VN in $\hat{V}_{r,2}$ and has the largest GRC;
    move the selected VN v_r from $\hat{V}_{r,1}$ to $\hat{V}_{r,2}$;
  else
    $f = 0$;
  end
end

within $[100, 150]$ units and $[80, 100]$ units, respectively. The DP of vSDN requests are also created by the GT-ITM tool, with a VN connectivity ratio of 0.5. We generate several sets of vSDN requests, and in each set, there are 5 requests whose numbers of VNs are the same. Both the TCAM demand of each VN and the bandwidth demand of each VL are randomly set within $[10, 15]$ units. We set $\theta = 0.5$ in Eq. (1), and test $\hat{H} \in \{1, 2, 3\}$ in the simulations. Fig. 4 shows the ILP's results on average resource cost per vSDN. It is interesting to notice that for a larger $\hat{H}$, the cost result from the ILP can be smaller, especially for vSDNs with relatively large topologies. This is because for a larger $\hat{H}$, the solution space becomes larger and since the ILP can solve the vSDN embedding problem exactly, it might find a better solution.

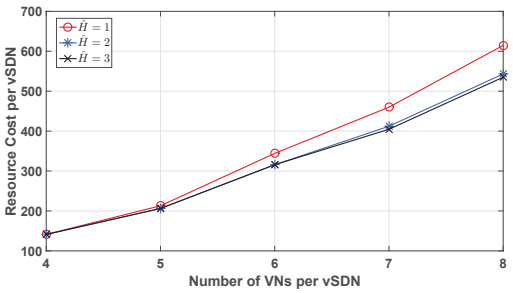

Fig. 4. Results from ILP with the small substrate network.

Then, we compare the performance of the proposed heuristic

TABLE I
COMPARISON OF AVERAGE RESOURCE COST PER VSDN

	$\hat{H}=1$					$\hat{H}=2$					$\hat{H}=3$				
# of VNs	2	3	4	5	6	5	6	7	8	9	8	9	10	11	12
ILP	36.2	64.0	108.8	162.4	334.6	155.2	195.4	225.4	248.8	335.6	214.6	298.2	302.4	353.2	378.2
Ours	36.2	64.0	115.0	186.6	362.2	173.4	228.8	286.2	369.4	421.4	318.8	379.0	356.2	444.2	454.8
Benchmark	36.2	64.0	121.0	205.0	402.4	174.0	246.8	319.0	409.6	545.4	338.2	472.8	512.0	644.8	693.2

and ILP. Also, to further verify the effectiveness of our proposed algorithms, a benchmark algorithm that uses the GRC-VNE in [9] to replace *Algorithm* 2 in *Algorithm* 1 is introduced. We name the benchmark algorithm as Benchmark and our proposed one in *Algorithm* 1 as Ours. Table I illustrates the average resource cost per vSDN from the algorithms for $\hat{H} \in \{1, 2, 3\}$. It can be seen that the ILP always provides the smallest resource cost, which is followed by Ours, and Benchmark always performs the worst. These results verify our analysis that since a vSDN consumes TCAM not only on the SNs that its VNs are embedded on but also on the SNs that are the intermediate nodes on the substrate paths to carry its VLs, we should try to map a VL to a relatively short substrate path to save TCAM resources.

Next, we evaluate the algorithms in a much larger substrate network. Due to the complexity of the ILP, we only compare Ours and Benchmark in this scenario. The substrate topology including 9 subnets, each of which includes 12 SNs with an SN connectivity ratio of 0.45. In each set, there are 10 requests whose numbers of VNs are the same, and all the other parameters are similar as before. We set $\hat{H} = 4$ and plot the results on average resource cost per vSDN in Fig. 5. It can be seen that Ours still outperforms Benchmark in terms of resource cost, and the performance gap between them becomes more significant when for larger vSDNs topologies.

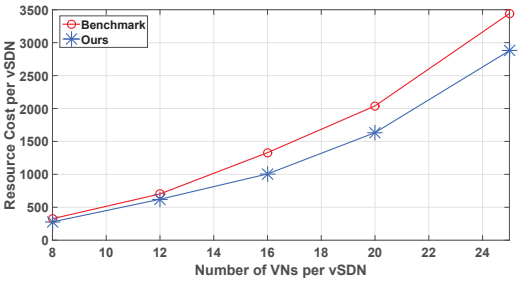

Fig. 5. Results from simulations with the large substrate network.

VI. CONCLUSION

We studied how to embed vSDNs cost-effectively over a substrate network with distributed NVHs for vDC formulation. Specifically, we designed an ILP model and a heuristic to jointly optimize the embedding schemes of the CP and DP of each vSDN, *i.e.*, to minimize the DP's resource consumption and limit the number of NVHs used in the CP simultaneously. Simulation results suggested that our proposed algorithms could embed vSDNs cost-effectively and significantly outperform the existing scheme based on GRC.

ACKNOWLEDGMENTS

This work was supported in part by the NSFC Project 61371117, Natural Science Research Project for Universities in Anhui (KJ2014ZD38), and the Strategic Priority Research Program of the CAS (XDA06011202).

REFERENCES

[1] P. Lu *et al.*, "Highly-efficient data migration and backup for big data applications in elastic optical inter-datacenter networks," *IEEE Netw.*, vol. 29, pp. 36–42, Sept./Oct. 2015.
[2] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *IEEE Comput.*, vol. 38, pp. 34–41, Apr. 2005.
[3] L. Gong and Z. Zhu, "Virtual optical network embedding (VONE) over elastic optical networks," *J. Lightw. Technol.*, vol. 32, pp. 450–460, Feb. 2014.
[4] L. Gong, H. Jiang, Y. Wang, and Z. Zhu, "Novel location-constrained virtual network embedding (LC-VNE) algorithms towards integrated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 24, pp. 3648–3661, Dec. 2016.
[5] H. Jiang, Y. Wang, L. Gong, and Z. Zhu, "Availability-aware survivable virtual network embedding (A-SVNE) in optical datacenter networks," *J. Opt. Commun. Netw.*, vol. 7, pp. 1160–1171, Dec. 2015.
[6] J. Yin *et al.*, "On-demand and reliable vSD-EON provisioning with correlated data and control plane embedding," in *Proc. of GLOBECOM 2016*, pp. 1–6, Dec. 2016.
[7] Z. Zhu *et al.*, "Demonstration of cooperative resource allocation in an OpenFlow-controlled multidomain and multinational SD-EON testbed," *J. Lightw. Technol.*, vol. 33, pp. 1508–1514, Apr. 2015.
[8] A. Fischer *et al.*, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, pp. 1888–1906, Fourth Quarter 2013.
[9] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. of INFOCOM 2014*, pp. 1–9, Apr. 2014.
[10] M. Zeng, W. Fang, and Z. Zhu, "Orchestrating tree-type VNF forwarding graphs in inter-DC elastic optical networks," *J. Lightw. Technol.*, vol. 34, pp. 3330–3341, Jul. 2016.
[11] B. Zhao, X. Chen, J. Zhu, and Z. Zhu, "Survivable control plane establishment with live control service backup and migration in SD-EONs," *J. Opt. Commun. Netw.*, vol. 8, pp. 371–381, Jun. 2016.
[12] X. Chen *et al.*, "Leveraging master-slave openflow controller arrangement to improve control plane resiliency in SD-EONs," *Opt. Express*, vol. 23, pp. 7550–7558, Mar. 2015.
[13] M. Demirci and M. Ammar, "Design and analysis of techniques for mapping virtual networks to software-defined network substrates," *Comput. Commun.*, vol. 45, pp. 1–10, Mar. 2014.
[14] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on network virtualization hypervisors for software defined networking," *IEEE Commun. Surveys Tuts.*, vol. 18, pp. 655–685, First Quarter 2016.
[15] P. Berde *et al.*, "ONOS: towards an open, distributed SDN OS," in *Proc. of ACM HotSDN 2014*, pp. 1–6, Aug. 2014.
[16] T. Koponen *et al.*, "Network virtualization in multi-tenant datacenters," in *Proc. of OSDI 2014*, pp. 203–216, Apr. 2014.
[17] A. Al-Shabibi *et al.*, "OpenVirteX: Make your virtual SDNs programmable," in *Proc. of ACM HotSDN 2014*, pp. 25–30, Aug. 2014.
[18] N. Chowdhury, M. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. of INFOCOM 2009*, pp. 783–791, Apr. 2009.
[19] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. of INFOCOM 1996*, pp. 594–602, May 1996.