On the Bilevel Optimization for Remapping Virtual Networks in an HOE-DCN

Hao Yang, Xiaoqin Pan, Sicheng Zhao, Binjie Ge, Hang Yu and Zuqing Zhu, *Senior Member, IEEE*

***Abstract*—Hybrid optical/electrical datacenter network (HOE-DCN) uses the inter-rack networks that consist of both electrical Ethernet switches and optical cross-connects (OXCs), for better cost-efficiency and scalability. Meanwhile, to provision dynamic network services well, the operator of an HOE-DCN needs to deploy virtual networks (VNTs) and remap them adaptively. Therefore, this work studies the problem of VNT remapping in an HOE-DCN from a novel perspective, *i.e.*, the remapping schemes should be optimized for not only the network status after the remapping but also the transition to realize it. Specifically, we model this problem as a bilevel optimization, where the upper-level optimization aims at selecting proper virtual machines (VMs) to migrate such that the estimated latency of VM migration can be minimized, and the lower-level optimization determines the actual scheme of VNT remapping for minimizing the number of resource hot-spots. We first formulate a bilevel mixed integer linear programming (BMILP) model for the bilevel optimization, and then propose a polynomial time algorithm based on enumeration to solve it approximately. Extensive simulations verify the effectiveness of our proposal.**

***Index Terms*—Datacenter networks, Virtual network remapping, Bilevel optimization, Approximation algorithm.**

I. Introduction

RECENTLY, the rapid development of network services, especially the data-/bandwidth-intensive ones, has stimulated the global deployment of datacenters (DCs) [1], and DC-related traffic already contributes the largest to Internet traffic [2, 3]. Hence, the infrastructure of DC networks (DCNs) is facing intimidating challenges [4], some of which are mainly due to the fact that traditional DCNs only consider electrical packet switching (EPS) [5, 6]. For instance, to reduce the capital expenditure (CAPEX), the oversubscription ratio in an EPS-based inter-rack network usually exceeds one, which might lead to bottlenecks for inter-rack communications [5]. The aforementioned challenges can potentially be addressed by introducing optical circuit switching (OCS) and integrating it with EPS in the inter-rack networks of DCNs [5, 6]. Such a DCN is normally referred to as a hybrid optical/electrical datacenter network (HOE-DCN), which can be more scalable than traditional DCNs [7, 8]. This is because an HOE-DCN integrates the benefits of EPS and OCS. Specifically, OCS provides larger bandwidth capacity and higher energy efficiency than EPS [9–11], while EPS outperforms OCS in terms of path setup latency and switching granularity [12].

It is known that a DCN usually supports network services by deploying virtual machines (VMs) in servers and setting up inter-rack connections to bridge the communications among VMs [13]. Therefore, each network service can be modeled as a virtual network (VNT), where its VMs are the virtual nodes (VNs) and the connections among the VMs are the virtual links (VLs). For example, a network service of Hadoop MapReduce [14] usually organizes its VMs as a cluster-type VNT, to accomplish computing tasks. Hence, if we consider the DCN as a substrate network (SNT), how to provision network services in it is equivalent to the problem of virtual network embedding (VNE) [15, 16].

However, network services can be established, readjusted, and terminated on-the-fly and active network services normally use IT and bandwidth resources dynamically [17]. This can progressively degrade the optimality of the initial VNE scheme of each network service, and thus makes it necessary to re-optimize the VNE scheme through VNT remapping [18–20]. Specifically, the operator can predict/estimate future network status periodically, and VNT remapping will be invoked if necessary [7, 8]. Note that, VNT remapping is generally more sophisticated than VNE, because, to maintain the operational complexity, proper VNs and VLs should be selected to reconfigure and the remapping of these virtual elements is restricted by their existing embedding schemes. Moreover, the VNT remapping in an HOE-DCN is intrinsically different from and more complex than that in traditional DCNs [18], which can be justified by looking at the example in Fig. 1.

The HOE-DCN in Fig. 1(a) takes the type of topologies that are considered in this work. The server racks provide the pool of IT resources for deploying VMs, and each of them is equipped with a top-of-rack (ToR) switch. The ToR switches are interconnected with both the EPS-based and OCS-based inter-rack networks. Specifically, the Ethernet switches (*i.e.*, the aggregation and core switches in Fig. 1(a)) are organized in a spine-and-leaf topology (*e.g.*, the classic fat-tree [21]) with oversubscription to build the EPS-based inter-rack network, while the OCS-based inter-rack network is based on an optical cross-connect (OXC)[1]. With OCS, the OXC can only support one-to-one connectivity between inputs and outputs, as shown in Fig. 1(a). Therefore, if VNT remapping needs to move VLs to/from the optical connections through the OXC, the OXC

H. Yang, X. Pan, S. Zhao, and Z. Zhu are with the School of Information Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China (Email: zqzhu@ieee.org).

H. Yang is also with the Department of Information Engineering, Southwest University of Science and Technology, Mianyang, Sichuan 621010, China.

X. Pan is also with the Engineering Technology Center, Southwest University of Science and Technology, Mianyang, Sichuan 621010, China.

B. Ge and H. Yu are with College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, Guangdong 518060, China.

Manuscript received on April 29, 2021.

[1]Note that, a commercially-available large-scale OXC can provide enough ports to interconnect hundreds of ToR switches, *e.g.*, a configuration of 384×384 ports is feasible for the one in [22].

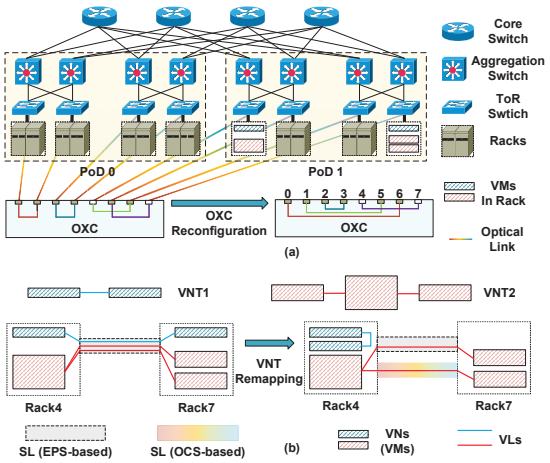


Fig. 1. Background on HOE-DCN, (a) Network architecture, and (b) Example of VNT remapping in it.

might be reconfigured and thus the overall topology of the HOE-DCN will be changed (*e.g.*, the VNT remapping in Fig. 1(b)). To this end, we can see that the VNT remapping in an HOE-DCN can change the SNT's topology as well, while this will not happen in a traditional DCN or most of the VNT remapping problems considered in the existing studies.

Previously, in [23, 24], we have studied the problem of VNT remapping in an HOE-DCN to re-balance the IT resource usage in racks. Nevertheless, the studies in [23, 24] should still be improved for the following two reasons. Firstly, as VNTs consume both IT and bandwidth resources, the VNT remapping should re-balance their usages jointly, such that resource "hot-spots" [7] in the HOE-DCN can be minimized and the quality-of-service (QoS) of network services can be ensured from multiple perspectives. Secondly and more importantly, VNT remapping involves VM migration and VL reconfiguration, between which the former is much more time-consuming and thus determines the overall latency of VNT remapping [25], while the studies in [23, 24] did not consider the potential latency of the resulting VM migration.

To avoid service interruption, the operator usually leverages live VM migration [26] to maintain the running of network services during VNT remapping. However, since a live migration normally needs to transfer several or even tens of gigabytes (GB) of data between two servers [17], it can occupy a fair amount of bandwidth consistently for a while [26], which might not only hinder normal service traffic but also cause network instability. Hence, the latency of VM migration should be regarded as the major cost of VNT remapping, and as the latency depends on the actual scheme of VNT remapping, it is desired to obtain the VNT remapping scheme that leads to the shortest latency of VM migration [25, 27].

This motivates us to revisit the problem of VNT remapping in an HOE-DCN in this work, and we would like to formulate a brand-new optimization to minimize the number of resource hot-spots and the estimated latency of VM migration simultaneously. Nevertheless, minimizing the number of resource hot-spots can make the VNT remapping migrate more VMs and prolong the latency of VM migration, and *vice versa*. Therefore, the two minimizations contradict to each other and can hardly be modeled with a single-level optimization.

In this work, we decide to model the VNT remapping as a bilevel optimization [28]. Specifically, the upper-level optimization aims at selecting proper VMs to migrate such that the estimated latency of VM migration can be minimized, while the lower-level optimization determines the actual scheme of VNT remapping for minimizing the number of resource hot-spots. In this bilevel model, the solution of the upper-level problem limits the set of feasible solutions for the lower-level one, while the lower-level solution evaluates that of the upper-level one. We first formulate a bilevel mixed integer linear programming (BMILP) model for the bilevel optimization, and design an exact algorithm that runs in exponential time. Then, we propose a polynomial time algorithm based on enumeration to solve the BMILP model approximately. Finally, extensive simulations verify the effectiveness of our proposal.

The rest of the paper is organized as follows. We survey the related work briefly in Section II. Section III provides the problem description and formulates the BMILP model. In Section IV, we first discuss the exact algorithm and then propose the polynomial time algorithm based on enumeration. The simulations for performance evaluations are discussed in Section V. Finally, Section VI summarizes the paper.

II. Related Work

As the fundamental problem to facilitate network virtualization, VNE has been studied for different types of networks in the literature [15, 16, 29–31]. Among them, a few studies have considered a DCN as the SNT [30, 31]. However, since network environment is usually dynamic, one-shot VNE can hardly be sufficient, and the usages of IT and bandwidth resources should be re-balanced frequently to keep the optimality of VNE [32]. Hence, VNT remapping should be studied to maintain the QoS of VNTs, and the existing investigations in this area covered not only algorithm designs [18–20] but also system implementations [33–35]. Nevertheless, none of the studies mentioned above has considered an HOE-DCN as the SNT, and thus their approaches cannot be leveraged to tackle the problem considered in this work, due to the unique feature of OXC (*i.e.*, its one-to-one connectivity makes the topology of the SNT changeable through VNT remapping).

Meanwhile, considering the dynamic nature of DCNs, numerous previous studies have been devoted to optimizing the procedure of VM migration such that the overall latency can be minimized [36–38], because VM migration is costly in DCNs and if not optimized, it can severely affect the QoS of network services [25]. However, these studies only tried to optimize the data transfers of VM migration according to a given VNT remapping plan (*i.e.*, where to migrate the VMs is predetermined). Recently, the authors of [39, 40] have proposed algorithms to tackle VNT remapping in consideration of the cost due to the latency of VM migration, but their investigation were based on the traditional EPS-based DCNs.

The idea of HOE-DCN was proposed more than a decade ago [5]. Lately, with the momentum gained from artificial intelligence, people have demonstrated effective network automation in an HOE-DCN by leveraging deep learning (DL),

such that the EPS-/OCS-based inter-rack networks can be orchestrated better for application-aware service provisioning [4, 7, 8]. Nevertheless, they only focused on the system designs and experimental demonstrations of DL-assisted network automation, but did not develop algorithms to optimize the VNT remapping in an HOE-DCN. With an over-simplified objective, we have studied the VNT remapping in an HOE-DCN in [23, 24], but as we have already explained, the optimization model should be further improved to consider more practical factors. Therefore, to the best of our knowledge, this work is first one that tries to optimize the VNT remapping in an HOE-DCN to minimize the number of resource hot-spots and the estimated latency of VM migration simultaneously.

TABLE I
MAJOR ABBREVIATIONS

Abbrev.	Full Name	Abbrev.	Full Name
SNT	Substrate network	VNT	Virtual network
SN	Substrate node	VNs	Virtual node
SL	Substrate link	VL	Virtual link
ToR	Top-of-rack	PoD	Points-of-delivery
VM	Virtual machine	LR	Lagrangian relaxation
LP	Linear programming	KKT	Karush-Kuhn-Tucker
HOE-DCN	Hybrid optical/electrical datacenter network		
EPS	Electrical packet switching		
OCS	Optical circuit switching		
OXC	Optical cross-connect		
ILP	Integer linear programming		
MILP	Mixed integer linear programming		
BMILP	Bilevel mixed integer linear programming		

III. PROBLEM DESCRIPTION

In this section, we first explain the network model of the VNT remapping in an HOE-DCN, and then formulate an BMILP model to describe the bilevel optimization for it. Table I lists the abbreviations that are frequently used in this paper.

A. Network Model

In our problem of VNT remapping, the SNT is an HOE-DCN and we model its topology as a graph $G(V_s, E_s)$, where V_s and E_s are the sets of substrate nodes (SNs) and substrate links (SLs), respectively. Here, each SN $v_s \in V_s$ is a server rack, which consists of a ToR switch and a server pool.

The IT resources on each server pool are CPU cycles and memory space, and we denote their capacities as C_{CPU} and C_{MEM}, respectively. Meanwhile, the total I/O capacity of each server pool is C_{BW}. Each ToR switch connects to both the EPS-based inter-rack network and an OXC, and we denote its total bandwidth capacities to/from the EPS-based and OCS-based inter-rack networks as B_{E} and B_{O}, respectively. Note that, as we assume that the EPS-based inter-rack network uses oversubscription, we have $C_{\text{BW}} > B_{\text{E}}$ in this work and the oversubscription ratio can be represented by $\frac{C_{\text{BW}}}{B_{\text{E}}}$. Meanwhile, the one-to-one connectivity of OXC restricts that at a given time, each ToR switch can only be connected with one other ToR switch through an OCS-based SL.

For each network service in the HOE-DCN, its VMs and the network connections among them formulate a VNT. As the VNT remapping considered in this work can reconfigure multiple VNTs, we define two sets (*i.e.*, V_r and E_r) to denote all the VMs and virtual links (VLs) of active VNTs, respectively. Note that, the dynamic nature of each network service makes the resource demands of the VMs and VLs in its VNT time-varying. Therefore, to adapt to future network status, we can estimate/forecast the resource demands (*e.g.*, the average or peak values) for a future period [7, 8], and remap the VNT accordingly (if necessary). Specifically, each VM $v_r \in V_r$ runs computing tasks and will consume certain CPU cycles and memory space in the future period, which are denoted as $c_{v_r}^{\text{CPU}}$ and $c_{v_r}^{\text{MEM}}$, respectively, while a VL $(v_r, u_r) \in E_r$ connects two VMs ($v_r, u_r \in V_r$), and its future bandwidth demand is defined as $b_{(v_r,u_r)}$. In a practical HOE-DCN, the estimation/prediction of future resource demands can be accomplished by the control plane [7, 8].

The uplink/downlink bandwidth demands of a ToR switch are the sums of the bandwidth demands of all the VLs that are from/to the VMs in its server pool. If the total IT or bandwidth demand of the VMs on a server pool approaches to the corresponding capacity, the network system will become highly-loaded such that the QoS of the related network services will be affected (*e.g.*, their job completion time can be prolonged [7, 8]). Hence, we define two thresholds, which tell the upper-limits of normal resource utilizations on server pools and EPS-based ports, as η_{IT} and η_{BW}, respectively. In other words, if the resource utilization on the server pool/EPS-based ports of a ToR switch exceeds η_{IT}/η_{BW}, the server pool/EPS-based ports will be a resource hot-spot in the HOE-DCN [7, 8]. Therefore, the VNT remapping should minimize resource hot-spots.

The latency of VM migration contributes the most to the cost of VNT remapping in DCNs [25]. Here, we hope to point out that even though the latency of VM migration can be reduced by leveraging a sophisticated algorithm to schedule the related data transfers [37], it is still relevant to consider the latency of VM migration when planning the VNT remapping. This is because the actual scheme of VNT remapping (*i.e.*, where to migrate the VMs) determines the lower-bound of the latency of VM migration. Hence, although the exact value of the latency can only be obtained by applying the VM migration scheduling algorithm, which is out of the scope of this work, we can estimate an approximation value for it by simply looking at the current network status. For instance, the latency of VM migration can be estimated by assuming that all the VMs are migrated in parallel, *i.e.,* their migrations start at the same time and share the available bandwidth evenly. Then, the estimated latency can be used as a performance indicator to evaluate the quality of a VNT remapping scheme.

Fig. 2 gives an example to explain why the estimated latency of VM migration should be considered when planning the VNT remapping. The network status in Fig. 2(a) indicates that the memory usages in *Racks* 1 and 3 will exceed the available memory spaces due to two VMs whose image size is 12 units. Then, because *Racks* 2 and 4 both have enough memory and bandwidth resources to accommodate the VMs, the VNT remapping schemes in Figs. 2(b) and 2(c) are equivalent from the perspective of minimizing the number of resource hot-spots. However, the conclusion will be different if we take the estimated latency of VM migration into account. Specifically,

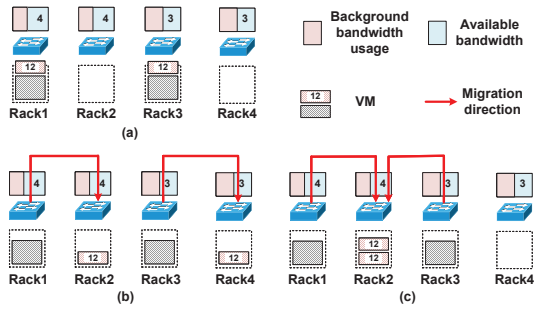


Fig. 2. Example on how the scheme of VNT remapping affects the latency of VM migration, (a) Network status before VNT remapping, (b) First VNT remapping scheme, and (c) Second VNT remapping scheme.

the VM migration latency of the scheme in Fig. 2(b) is $\max(\frac{12}{4}, \frac{12}{3}) = 4$ time-units, while that of the scheme in Fig. 2(c) is $\frac{24}{4} = 6$ time-units. The bandwidth bottleneck to *Rack* 2 will prolong the latency of VM migration, no matter how we schedule the related data transfers. Therefore, the VNT remapping scheme in Fig. 2(b) is more preferable. To this end, we can see that for the VNT remapping, we should minimize the number of resource hot-spots and the estimated latency of VM migration simultaneously.

B. Bilevel Optimization Model

As the two aforementioned minimizations have mutual restraint, we artificially divide the decision maker that plans VNT remapping into two logical entities, *i.e.*, the VM picker and remapping planner, such that the VNT remapping problem can be modeled as a bilevel optimization. Here, the VM picker considers the upper-level optimization to select the VMs to migrate for minimizing the resulting VM migration latency, while the remapping planner is in charge of the lower-level optimization to finalize the VNT remapping scheme (*i.e.*, where to migrate the selected VMs and how to reconfigure the OXC) for minimizing the number of resource hot-spots. Hence, neither the VM picker nor the remapping planner can solve its optimization independently. This also justifies why the problem of VNT remapping in an HOE-DCN should not be modeled as a single-level optimization. We formulate the following BMILP to describe the bilevel model. Table II lists the parameters of the upper- and lower-level optimizations.

1) Upper-level Optimization: It is for the VM picker to select the VMs to migrate, such that the resulting VM migration latency can be minimized. Table III summarizes its variables.

Objective:

The upper-level optimization needs to minimize the latency of VM migration, which is estimated in the lower-level one.

$$\text{Minimize} \quad T. \tag{1}$$

Constraints:

$$\begin{cases} \sum\limits_{v_r \in V_r} P_{v_r,v_s} \cdot K_{v_r} \cdot c_{v_r}^{\text{CPU}} \le C_{\text{CPU}}, \\ \sum\limits_{v_r \in V_r} P_{v_r,v_s} \cdot K_{v_r} \cdot c_{v_r}^{\text{MEM}} \le C_{\text{MEM}}, \end{cases} \quad \forall v_s \in V_s. \tag{2}$$

Eq. (2) ensures that on each SN, the total IT resource usage of the VMs, which are not selected for migration, does not exceed the IT resource capacity of its server pool.

$$\begin{cases} \sum\limits_{(v_r,u_r) \in E_r} P_{v_r,v_s} \cdot K_{v_r} \cdot b_{(v_r,u_r)} \le C_{\text{BW}}, \\ \sum\limits_{(v_r,u_r) \in E_r} P_{u_r,v_s} \cdot K_{u_r} \cdot b_{(v_r,u_r)} \le C_{\text{BW}}, \end{cases} \quad \forall v_s \in V_s. \tag{3}$$

Eq. (3) ensures that on each SN, the total bandwidth from/to the VMs, which are not selected for migration, does not exceed the corresponding bandwidth capacity of its server pool.

The constraints in Eqs. (2) and (3) are introduced because $c_{v_r}^{\text{CPU}}$, $c_{v_r}^{\text{MEM}}$, and $b_{(v_r,u_r)}$ are actually the estimated/forecasted resource demands for a future period. In other words, we need to ensure that the future resource demands of remaining VMs on each rack will not exceed the corresponding capacities.

2) Lower-level Optimization: It is for the remapping planner to determine the VNT remapping scheme based on the selected VMs, such that the number of resource hot-spots can be minimized. As the VMs for migration are selected by the upper-level optimization, $\{K_{v_r}\}$ become parameters here. We list the variables to optimize in Table IV.

Objective:

The objective is to minimize the resource hot-spots.

$$\text{Minimize} \quad \sum_{v_s \in V_s} \left(g_{v_s}^{\text{IT}} + g_{v_s}^{\text{EO}} + g_{v_s}^{\text{EI}} \right). \tag{4}$$

TABLE II
COMMON PARAMETERS

Substrate Network (SNT)	
$G(V_s, E_s)$	The topology of the SNT (*i.e.*, an HOE-DCN).
C_{CPU}	The capacity of CPU cycles on each rack's server pool.
C_{MEM}	The total memory space on each rack's server pool.
C_{BW}	The total I/O capacity of each rack's server pool.
B_{E}	The total bandwidth capacity on each ToR switch to/from the EPS-based inter-rack network.
$B_{v_s}^{\text{EO}}/B_{v_s}^{\text{EI}}$	The available bandwidth on the ToR switch in SN $v_s \in V_s$ to/from the EPS-based inter-rack network.
P_{v_r,v_s}	The binary that equals 1 if VM v_r is embedded on SN v_s before VNT remapping, and 0 otherwise.
L_{v_s,u_s}	The binary that equals 1 if the ToR switches of SNs v_s and u_s are connected through the OXC before VNT remapping, and 0 otherwise.
$\eta_{\text{IT}}/\eta_{\text{BW}}$	The threshold for identifying resource hot-spots, *i.e.*, the upper-limit of normal resource utilization on the server pool/EPS-based ports of a ToR switch.
Virtual Network (VNT)	
V_r/E_r	The set of VMs/VLs in active VNTs.
$c_{v_r}^{\text{CPU}}$	The CPU demand of VM $v_r \in V_r$.
$c_{v_r}^{\text{MEM}}$	The memory demand of VM $v_r \in V_r$.
S_{v_r}	The amount of data transfer to migrate VM v_r.
$b_{(v_r,u_r)}$	The bandwidth demand of VL $(v_r, u_r) \in E_r$.
Auxiliary Parameters	
W	A large positive constant.

TABLE III
UPPER-LEVEL VARIABLES

K_{v_r}	The binary variable that equals 1 if VM v_r stays on its current SN (is not migrated) after VNT remapping, and 0 otherwise.
T	The estimated latency of VM migration.

TABLE IV
LOWER-LEVEL VARIABLES

$P^*_{v_r,v_s}$	The binary variable that equals 1 if VM v_r is mapped on SN v_s after VNT remapping, and 0 otherwise.
$L^*_{v_s,u_s}$	The binary variable that equals 1 if the ToR switches of SNs v_s and u_s are connected through the OXC after VNT remapping, and 0 otherwise.
$x^{v_s,u_s}_{(v_r,u_r)}$	The binary variable that equals 1 if VL (v_r,u_r) is from v_s to u_s after VNT remapping, and 0 otherwise.
$y^{v_s,u_s}_{(v_r,u_r)}$	The binary variable that equals 1 if VL (v_r,u_r) is from v_s to u_s and goes through the OXC after VNT remapping, and 0 otherwise.
$z^{v_s,u_s}_{(v_r,u_r)}$	The binary variable that equals 1 if VL (v_r,u_r) is from v_s to u_s and goes through the EPS-based inter-rack network after VNT remapping, and 0 otherwise.
$g^{\text{EO}}_{v_s}/g^{\text{EI}}_{v_s}$	The binary variable that equals 1 if the ToR switch of SN v_s encounters a resource hot-spot in the uplink/downlink direction of its EPS-based inter-rack network after VNT remapping, and 0 otherwise.
$g^{\text{IT}}_{v_s}$	The binary variable that equals 1 if SN v_s encounters a resource hot-spot in its server pool, and 0 otherwise.

Constraints:

$$K_{v_r} = \sum_{v_s \in V_s} P_{v_r,v_s} \cdot P^*_{v_r,v_s}, \quad \forall v_r \in V_r. \tag{5}$$

Eq. (5) ensures that only the VMs selected by the upper-level optimization can be migrated.

$$\sum_{v_s \in V_s} P^*_{v_r,v_s} = 1, \quad \forall v_r \in V_r. \tag{6}$$

Eq. (6) ensures that each VM is embedded on one SN.

$$\begin{cases} \sum\limits_{v_r \in V_r} P^*_{v_r,v_s} \cdot c^{\text{CPU}}_{v_r} \le C_{\text{CPU}}, \\ \sum\limits_{v_r \in V_r} P^*_{v_r,v_s} \cdot c^{\text{MEM}}_{v_r} \le C_{\text{MEM}}, \\ \sum\limits_{(v_r,u_r) \in E_r} P^*_{v_r,v_s} \cdot b_{(v_r,u_r)} \le C_{\text{BW}}, \\ \sum\limits_{(v_r,u_r) \in E_r} P^*_{u_r,v_s} \cdot b_{(v_r,u_r)} \le C_{\text{BW}}, \end{cases} \quad \forall v_s \in V_s. \tag{7}$$

Eq. (7) ensures that the resource constraints are still satisfied.

$$\sum_{u_s \in V_s \setminus v_s} L^*_{v_s,u_s} = 1, \quad \forall v_s \in V_s. \tag{8}$$

Eq. (8) ensures that each ToR switch can only be connected with one other ToR switch through the OXC.

$$L^*_{v_s,u_s} = L^*_{u_s,v_s}, \quad \{v_s,u_s : v_s \ne u_s,\ v_s,u_s \in V_s\}. \tag{9}$$

Eq. (9) ensures that each OCS-based connection between a pair of ToR switches is bidirectional.

$$\begin{cases} x^{v_s,u_s}_{(v_r,u_r)} \ge P^*_{v_r,v_s} + P^*_{u_r,u_s} - 1, \\ x^{v_s,u_s}_{(v_r,u_r)} \le \frac{1}{2} \cdot \left(P^*_{v_r,v_s} + P^*_{u_r,u_s}\right), \end{cases} \forall (v_r,u_r) \in E_r, \forall v_s,u_s \in V_s. \tag{10}$$

Eq. (10) ensures that if VMs v_r and u_r are mapped on SNs v_s and u_s, respectively, VL (v_r,u_r) is embedded on an SL between SNs v_s and u_s accordingly.

$$y^{v_s,u_s}_{(v_r,u_r)} \le \frac{1}{3} \cdot \left(P^*_{v_r,v_s} + P^*_{u_r,u_s} + L^*_{v_s,u_s}\right), \quad \forall (v_r,u_r) \in E_r,\ \forall v_s,u_s \in V_s. \tag{11}$$

Eq. (11) ensures that if a VL is embedded on an SL between a pair of ToR switches that are connected through the OXC, it can be embedded on an OCS-based SL.

$$z^{v_s,u_s}_{(v_r,u_r)} + y^{v_s,u_s}_{(v_r,u_r)} = x^{v_s,u_s}_{(v_r,u_r)}, \ \forall (v_r,u_r) \in E_r,\ \forall v_s,u_s \in V_s. \tag{12}$$

Eq. (12) ensures that each VL is embedded on either an EPS-based SL or an OCS-based SL between a pair of ToR switches.

$$W \cdot g^{\text{EO}}_{v_s} \ge \left(\sum_{(v_r,u_r) \in E_r} \sum_{u_s \in V_s \setminus v_s} z^{v_s,u_s}_{(v_r,u_r)} \cdot b_{(v_r,u_r)} \right) - \eta_{\text{BW}} \cdot B_{\text{E}}, \quad \forall v_s \in V_s, \tag{13}$$

$$W \cdot g^{\text{EI}}_{v_s} \ge \left(\sum_{(v_r,u_r) \in E_r} \sum_{u_s \in V_s \setminus v_s} z^{u_s,v_s}_{(v_r,u_r)} \cdot b_{(v_r,u_r)} \right) - \eta_{\text{BW}} \cdot B_{\text{E}}, \quad \forall v_s \in V_s, \tag{14}$$

$$\begin{cases} W \cdot g^{\text{IT}}_{v_s} \ge \left(\sum\limits_{v_r \in V_r} P^*_{v_r,v_s} \cdot c^{\text{CPU}}_{v_r} \right) - \eta_{\text{IT}} \cdot C_{\text{CPU}}, \\ W \cdot g^{\text{IT}}_{v_s} \ge \left(\sum\limits_{v_r \in V_r} P^*_{v_r,v_s} \cdot c^{\text{MEM}}_{v_r} \right) - \eta_{\text{IT}} \cdot C_{\text{MEM}}, \end{cases} \quad \forall v_s \in V_s. \tag{15}$$

Eq. (13)-(15) ensure that after VNT remapping, all the resource hot-spots are identified correctly. Specifically, for a rack, if the total bandwidth demand in the uplink/downlink direction or the total CPU/memory demand exceeds the corresponding threshold, the related decision variable for resource hot-spot (*i.e.*, $g^{\text{EO}}_{v_s}$, $g^{\text{EI}}_{v_s}$, or $g^{\text{IT}}_{v_s}$) is set to 1.

$$\begin{cases} B^{\text{EO}}_{v_s} \cdot T \ge \sum\limits_{v_r \in V_r} \sum\limits_{u_s \in V_s \setminus v_s} P_{v_r,v_s} \cdot P^*_{v_r,u_s} \cdot (1 - L_{v_s,u_s}) \cdot S_{v_r}, \\ B^{\text{EI}}_{v_s} \cdot T \ge \sum\limits_{v_r \in V_r} \sum\limits_{u_s \in V_s \setminus v_s} P_{v_r,u_s} \cdot P^*_{v_r,v_s} \cdot (1 - L_{v_s,u_s}) \cdot S_{v_r}, \end{cases} \quad \forall v_s \in V_s. \tag{16}$$

Eq. (16) ensures that the latency of VM migration is estimated correctly, by assuming simultaneous migrations in parallel[2].

IV. ALGORITHM DESIGN

According to the analysis in [41], the bilevel optimization formulated above is $\mathcal{NP}$-hard. Hence, in this section, we first design an exact algorithm that runs in exponential time, and then propose a polynomial time approximation algorithm.

A. Exact Algorithm based On Enumeration

Since the BMILP formulated in Section III-B contains binary variables, the bilevel model does not have convexity [42]. Hence, we cannot leverage the commonly-used methods based on Karush-Kuhn-Tucker (KKT) conditions and strong duality [43] to transform it into a single-level optimization. Meanwhile, as the bilevel model has separable upper and lower levels, *i.e.*, the upper-level problem can be solved without considering the lower-level one [42], we can design an

[2]As each OCS-based SL normally has a much larger bandwidth capacity than an EPS-based SL, we only consider the available bandwidth in the EPS-based inter-rack network to estimate the latency of VM migration, for addressing the worst case scenario.

enumeration-based exact algorithm to solve it [44, 45]. Specifically, the exact algorithm first gets all the feasible solutions of the upper-level problem, then obtains the corresponding solutions of the lower-level one, and finally checks the solution combinations of the upper- and lower-level problems to find the optimal solution of the bilevel model.

Algorithm 1 describes the detailed procedure of the exact algorithm. It first obtains all the feasible solution of the upper-level problem under the resource constraints (*i.e.*, each solution is a feasible set of VMs for migration $\{K_{v_r}\}$). Then, for each solution, it formulates and solves a lower-level problem to obtain a solution of the bilevel model, *i.e.*, $\{\tilde{T}, \tilde{P}^*_{v_r,v_s}, \tilde{L}^*_{v_s,u_s}\}$ (*Line* 3). Next, we compare $\tilde{T}$ with the best-known latency of VM migration T^*, and update the best-known solution if we have $\tilde{T} < T^*$ (*Lines* 4-7). Finally, after all the feasible solutions of the upper-level problem have been checked, the optimal solution of the bilevel optimization can be obtained.

Algorithm 1: Exact Algorithm based on Enumeration

1 $T^* = +\infty$, $\{P^*_{v_r,v_s} = 0\}$, $\{L^*_{v_s,v'_s} = 0\}$;
2 **for** *each feasible solution* $\{K_{v_r}\}$ *of upper-level* **do**
3 formulate a lower-level MILP with $\{K_{v_r}\}$ and solve it to get $\tilde{T}$, $\{\tilde{P}^*_{v_r,v_s}\}$ and $\{\tilde{L}^*_{v_s,u_s}\}$;
4 **if** $\tilde{T} < T^*$ **then**
5 $T^* = \tilde{T}$, $\{P^*_{v_r,v_s}\} = \{\tilde{P}^*_{v_r,v_s}\}$;
6 $\{L^*_{v_s,u_s}\} = \{\tilde{L}^*_{v_s,u_s}\}$;
7 **end**
8 **end**
9 **return** $\{P^*_{v_r,v_s}\}$ and $\{L^*_{v_s,u_s}\}$;

B. Considerations for Approximation Algorithm Design

In *Algorithm* 1, neither the enumeration of all the feasible solutions of the upper-level problem nor the solving of the lower-level mixed integer linear programming (MILP) can be completed in polynomial time. Hence, it does not scale well, which motivates us to leverage its procedure to design a polynomial time approximation algorithm (*i.e.*, *Algorithm* 2).

C. Approximation Algorithm for Upper-level Problem

In *Algorithm* 2, we first try to improve the time-efficiency of the enumeration of the upper-level problem's solutions. Eq. (16) suggests that the latency of VM migration generally increases with the maximal total size of the VMs to/from a server pool. Hence, we develop an enumeration strategy to only check N most promising solutions of the upper-level problem, as shown in *Lines* 1-7 of *Algorithm* 2. The basic idea is to only select the "necessary" VMs for migration under a set of shrunk resource constraints, such that the total size of the selected VMs on each SN is minimized. *Line* 1 defines $\Delta\eta_{\text{IT}}$ and $\Delta\eta_{\text{BW}}$ as the steps for shrinking the IT and I/O capacities of each SN, respectively. Then, the for-loop that covers *Lines* 2-7 enumerates N most promising solutions, where *Line* 3 calculates the shrinking ratios ($\tilde{\eta}_{\text{IT}}$ and $\tilde{\eta}_{\text{BW}}$) for resource constraints in each iteration. Specifically, the iterations shrink the resource constraints from 100% to right below the corresponding thresholds ($\tilde{\eta}_{\text{IT}}$ and $\tilde{\eta}_{\text{BW}}$) for identifying resource hot-spots in the HOE-DCN.

With $\tilde{\eta}_{\text{IT}}$ and $\tilde{\eta}_{\text{BW}}$, *Line* 5 formulates the following optimization to tackle the upper-level problem for each SN $v_s \in V_s$.

$$\begin{aligned}
\text{Maximize} \quad & \sum_{\{v_r:P_{v_r,v_s}=1\}} K_{v_r}\cdot S_{v_r},\\
\text{s.t.} \quad & \sum_{\{v_r:P_{v_r,v_s}=1\}} K_{v_r}\cdot c^{\text{CPU}}_{v_r} < \tilde{\eta}_{\text{IT}}\cdot C_{\text{CPU}},\\
& \sum_{\{v_r:P_{v_r,v_s}=1\}} K_{v_r}\cdot c^{\text{MEM}}_{v_r} < \tilde{\eta}_{\text{IT}}\cdot C_{\text{MEM}},\\
& \sum_{\{v_r:P_{v_r,v_s}=1\}} K_{v_r}\cdot b^{\text{in}}_{v_r} < \tilde{\eta}_{\text{BW}}\cdot C_{\text{BW}},\\
& \sum_{\{v_r:P_{v_r,v_s}=1\}} K_{v_r}\cdot b^{\text{out}}_{v_r} < \tilde{\eta}_{\text{BW}}\cdot C_{\text{BW}}.
\end{aligned} \tag{17}$$

where $\{v_r : P_{v_r,v_s} = 1\}$ denotes the set of VMs that are embedded on SN v_s before VNT remapping, and $b^{\text{in}}_{v_r}$/$b^{\text{out}}_{v_r}$ is the total I/O resource demand to/from VM v_r (*i.e.*, the total bandwidth demand of the VLs that go in/out v_r).

If we treat $\tilde{\eta}_{\text{IT}}\cdot C_{\text{CPU}}$, $\tilde{\eta}_{\text{IT}}\cdot C_{\text{MEM}}$ and $\tilde{\eta}_{\text{BW}}\cdot C_{\text{BW}}$ as the capacities of a 4-dimensional (4D) knapsack, $\{c^{\text{CPU}}_{v_r}, c^{\text{MEM}}_{v_r}, b^{\text{in}}_{v_r}, b^{\text{out}}_{v_r}\}$ as the 4D size of an item (*i.e.*, a VM v_r), and S_{v_r} as the value of an item, the optimization in Eq. (17) actually constitutes a typical 4D knapsack problem. As the 4D knapsack is still $\mathcal{NP}$-hard, we design a polynomial time approximate algorithm based on Lagrangian relaxation (LR) to solve it in *Line* 6. We first build a dual problem, whose solution gives an upper-bound on the optimal solution of Eq. (17).

$$\begin{aligned}
\text{Minimize} \quad & \mathcal{L}_{\text{dual}}(\lambda_1,\lambda_2,\lambda_3) = \max_{\{K_{v_r}\}} \Bigg[\sum_{\{v_r:P_{v_r,v_s}=1\}} K_{v_r}\cdot S_{v_r}\\
& + \lambda_1\cdot\left(\tilde{\eta}_{\text{IT}}\cdot C_{\text{MEM}} - \sum_{\{v_r:P_{v_r,v_s}=1\}} K_{v_r}\cdot c^{\text{MEM}}_{v_r}\right)\\
& + \lambda_2\cdot\left(\tilde{\eta}_{\text{BW}}\cdot C_{\text{BW}} - \sum_{\{v_r:P_{v_r,v_s}=1\}} K_{v_r}\cdot b^{\text{in}}_{v_r}\right)\\
& + \lambda_3\cdot\left(\tilde{\eta}_{\text{BW}}\cdot C_{\text{BW}} - \sum_{\{v_r:P_{v_r,v_s}=1\}} K_{v_r}\cdot b^{\text{out}}_{v_r}\right)\Bigg],\\
\text{s.t.} \quad & \sum_{\{v_r:P_{v_r,v_s}=1\}} K_{v_r}\cdot c^{\text{CPU}}_{v_r} < \tilde{\eta}_{\text{IT}}\cdot C_{\text{CPU}}.
\end{aligned} \tag{18}$$

where λ_1, λ_2 and λ_3 are the non-negative Lagrangian multipliers. Eq. (18) can be further simplified as

$$\begin{aligned}
\text{Minimize} \quad & \mathcal{L}_{\text{dual}}(\lambda_1,\lambda_2,\lambda_3) = \max_{\{K_{v_r}\}} \Bigg\{ \Bigg[\sum_{\{v_r:P_{v_r,v_s}=1\}} K_{v_r}\cdot\\
& \left(S_{v_r} - \lambda_1\cdot c^{\text{MEM}}_{v_r} - \lambda_2\cdot b^{\text{in}}_{v_r} - \lambda_3\cdot b^{\text{out}}_{v_r}\right)\Bigg]\\
& + \lambda_1\cdot\tilde{\eta}_{\text{IT}}\cdot C_{\text{MEM}} + (\lambda_2+\lambda_3)\cdot\tilde{\eta}_{\text{BW}}\cdot C_{\text{BW}}\},\\
\text{s.t.} \quad & \sum_{\{v_r:P_{v_r,v_s}=1\}} K_{v_r}\cdot c^{\text{CPU}}_{v_r} < \tilde{\eta}_{\text{IT}}\cdot C_{\text{CPU}}.
\end{aligned} \tag{19}$$

Therefore, for specific λ_1, λ_2 and λ_3, the optimization in Eq. (19) can be solved by dynamic programming with the time complexity of $O\left(|\{v_r : P_{v_r,v_s} = 1\}|\cdot C_{\text{CPU}}\right)$ to get the

Algorithm 2: Approximation Algorithm for Bilevel Model

$\Delta\eta_{\text{IT}} = \frac{1-\eta_{\text{IT}}}{N-1}$, $\Delta\eta_{\text{BW}} = \frac{1-\eta_{\text{BW}}}{N-1}$;
for $i \in [0, N-1]$ **do**
  $\tilde{\eta}_{\text{IT}} = 1 - i \cdot \Delta\eta_{\text{IT}}$, $\tilde{\eta}_{\text{BW}} = 1 - i \cdot \Delta\eta_{\text{BW}}$;
  shrink resource constraints with $\tilde{\eta}_{\text{IT}}$ and $\tilde{\eta}_{\text{BW}}$;
  formulate an optimization with the shrunk constraints for the upper-level problem (as in Eq. (17));
  solve the optimization in Eq. (17) with *Algorithm* 4 to obtain a feasible solution of the upper-level problem;
end
$T^* = +\infty$, $\{P^*_{v_r,v_s} = 0\}$, $\{L^*_{v_s,v'_s} = 0\}$;
for *each feasible solution* $\{K_{v_r}\}$ *of upper-level* **do**
  formulate a lower-level MILP with $\{K_{v_r}\}$ and solve it with *Algorithm* 5 to get $\tilde{T}$, $\{\tilde{P}^*_{v_r,v_s}\}$ and $\{\tilde{L}^*_{v_s,u_s}\}$;
  if $\tilde{T} < T^*$ **then**
    $T^* = \tilde{T}$, $\{P^*_{v_r,v_s}\} = \{\tilde{P}^*_{v_r,v_s}\}$;
    $\{L^*_{v_s,u_s}\} = \{\tilde{L}^*_{v_s,u_s}\}$;
  end
end
return $\{P^*_{v_r,v_s}\}$ and $\{L^*_{v_s,u_s}\}$;

optimal solution $\{\tilde{K}_{v_r}\}$. However, $\{\tilde{K}_{v_r}\}$ might not be a feasible solution for the optimization in Eq. (17). Hence, we design *Algorithm* 3 to build a feasible solution $\{K_{v_r}\}$ based on $\{\tilde{K}_{v_r}\}$, whose time complexity is $O\left(|\{v_r : P_{v_r,v_s} = 1\}|\right)$.

Algorithm 3: Building Feasible Solution for Eq. (17)

Input: $\{\tilde{K}_{v_r}\}$
$\mathbf{S} = 0$, $c^{\text{CPU}} = 0$, $c^{\text{MEM}} = 0$, $b^{\text{in}} = 0$, $b^{\text{out}} = 0$, $flag = 0$;
for *each* v_r *in* $\{v_r : P_{v_r,v_s} = 1\}$ **do**
  if $flag = 1$ **then**
    $K_{v_r} = 0$, **continue**;
  end
  if $\tilde{K}_{v_r} = 0$ **then**
    $\mathbf{S} = \mathbf{S} + S_{v_r}$, $c^{\text{CPU}} = c^{\text{CPU}} + c^{\text{CPU}}_{v_r}$, $c^{\text{MEM}} = c^{\text{MEM}} + c^{\text{MEM}}_{v_r}$;
    $b^{\text{in}} = b^{\text{in}} + b^{\text{in}}_{v_r}$, $b^{\text{out}} = b^{\text{out}} + b^{\text{out}}_{v_r}$;
    if ($c^{CPU} \geq \tilde{\eta}_{IT} \cdot C_{CPU}$) *or* ($c^{MEM} \geq \tilde{\eta}_{IT} \cdot C_{MEM}$) *or* ($b^{in} \geq \tilde{\eta}_{BW} \cdot C_{BW}$) *or* ($b^{out} \geq \tilde{\eta}_{BW} \cdot C_{BW}$) **then**
      $K_{v_r} = 0$, $flag = 1$, **continue**;
    end
    $K_{v_r} = 1$;
  end
end
return $\{K_{v_r}\}$ and $\mathbf{S}$;

For specific λ_1, λ_2 and λ_3, we use $\{\tilde{K}_{v_r}\}$ to calculate $\mathcal{L}_{\text{dual}}(\lambda_1, \lambda_2, \lambda_3)$, which is an upper-bound of the solution of Eq. (17), while the $\mathbf{S}$ obtained by *Algorithm* 3 is a lower-bound of Eq. (17). Then, with the principle of LR, we can update the values of λ_1, λ_2 and λ_3 in iterations with the sub-gradient method in [46], such that the gap between the upper and lower-bounds is reduced continuously to make the best-known feasible solution approximate the optimal one. *Algorithm* 4 shows the overall procedure of the LR. *Line* 1 is for the initialization. The subsequent for-loop solves Eq. (17) with LR for all the server racks (*Lines* 2-21).

Specifically, for each server rack (*i.e.*, an SN $v_s \in V_s$), the while-loop that covers *Lines* 4-20 tries to improve the quality of the solution until the relative dual gap is smaller than a preset threshold γ_1. *Line* 5 solves Eq. (19) for $\mathcal{L}_{\text{dual}}(\lambda_1, \lambda_2, \lambda_3)$ and $\{\tilde{K}_{v_r}\}$, and then *Lines* 6-14 update the upper-bound ub with $\mathcal{L}_{\text{dual}}(\lambda_1, \lambda_2, \lambda_3)$ and modify the step-size coefficient w. Here, we use t to record the number of iterations for which ub has not been updated, and if it exceeds a preset threshold t_{TH}, we divide w by 2 (*Line* 10). *Line* 15 inputs $\{\tilde{K}_{v_r}\}$ to *Algorithm* 3 and gets a feasible solution of Eq. (17) ($\mathbf{S}$ and $\{K_{v_r}\}$). As Eq. (17) is for maximization, its feasible solution sets a lower-bound on the objective. Hence, in *Line* 16, we update the lower-bound lb with $\mathbf{S}$, and calculate the sub-gradient vectors of $\mathcal{L}_{\text{dual}}(\lambda_1, \lambda_2, \lambda_3)$ regarding λ_1, λ_2 and λ_3 as

$$\begin{cases} f(\lambda_1) = \dfrac{\partial \mathcal{L}_{\text{dual}}}{\partial \lambda_1} = \tilde{\eta}_{\text{IT}} \cdot C_{\text{MEM}} - \displaystyle\sum_{\{v_r : P_{v_r,v_s}=1\}} K_{v_r} \cdot c^{\text{MEM}}_{v_r}, \\ f(\lambda_2) = \dfrac{\partial \mathcal{L}_{\text{dual}}}{\partial \lambda_2} = \tilde{\eta}_{\text{BW}} \cdot C_{\text{BW}} - \displaystyle\sum_{\{v_r : P_{v_r,v_s}=1\}} K_{v_r} \cdot b^{\text{in}}_{v_r}, \\ f(\lambda_3) = \dfrac{\partial \mathcal{L}_{\text{dual}}}{\partial \lambda_3} = \tilde{\eta}_{\text{BW}} \cdot C_{\text{BW}} - \displaystyle\sum_{\{v_r : P_{v_r,v_s}=1\}} K_{v_r} \cdot b^{\text{out}}_{v_r}. \end{cases} \tag{20}$$

Line 17 obtains the step-size in the current iteration as [47]

$$\mu^n = \frac{w \cdot [\mathcal{L}_{\text{dual}}(\lambda^n_1, \lambda^n_2, \lambda^n_3) - \mathbf{S}]}{[f(\lambda^n_1) + f(\lambda^n_2) + f(\lambda^n_3)]^2}, \tag{21}$$

where λ^n_1, λ^n_2 and λ^n_3 represent the values of λ_1, λ_2 and λ_3 in the current iteration (*i.e.*, the n-th), respectively. We get the λ_1, λ_2 and λ_3 for the next iteration in *Line* 18, as

$$\begin{cases} \lambda^{n+1}_1 = \max\left[0, \lambda^n_1 - \mu^n \cdot f(\lambda^n_1)\right], \\ \lambda^{n+1}_2 = \max\left[0, \lambda^n_2 - \mu^n \cdot f(\lambda^n_2)\right], \\ \lambda^{n+1}_3 = \max\left[0, \lambda^n_3 - \mu^n \cdot f(\lambda^n_3)\right]. \end{cases} \tag{22}$$

Finally, *Algorithm* 4 solves the upper-level problem that uses a specific set of shrunk resource constraints in *Line* 22. As Eq. (17) transforms the minimization of the upper-level problem into a maximization, the approximation ratio of *Algorithm* 4 can be analyzed as follows. If we denote the optimal solution of Eq. (17) as $\mathbf{S}_{\text{ILP}}$, we have $\mathbf{S}_{\text{ILP}} \leq \mathcal{L}_{\text{dual}}(\lambda_1, \lambda_2, \lambda_3) = ub$ and $lb = \mathbf{S}$, where $\mathbf{S}$ is the solution obtained by *Algorithm* 4. Therefore, the approximate ratio ϵ_1 is

$$\epsilon_1 = \frac{\mathbf{S}}{\mathbf{S}_{\text{ILP}}} \geq \frac{\mathbf{S}}{\mathcal{L}_{\text{dual}}(\lambda_1, \lambda_2, \lambda_3)} = \frac{lb}{ub} > 1 - \gamma_1, \tag{23}$$

which confirms that the approximation ratio is at least $1 - \gamma_1$.

According to the principle of LR, *Algorithm* 4 runs in polynomial time. Hence, *Line* 6 in *Algorithm* 2 can be tackled in polynomial time as well. Then, as the enumeration strategy in *Lines* 1-7 of *Algorithm* 2 only considers N most promising solutions of the upper-level problem, it runs in polynomial time too. Next, the time complexity of *Lines* 8-15 of *Algorithm* 2 is dominated by that of *Line* 10 there. In the next subsection, we will design an approximation algorithm to solve the lower-level MILP in polynomial time (*i.e.*, for the *Algorithm* 5 in *Line* 10 of *Algorithm* 2). Finally, we can confirm that *Algorithm* 2 runs in polynomial time.

Algorithm 4: Lagrangian Relaxation to Solve Eq. (17)

Input: γ_1
$\{K_{v_r} = 0\}$;
for *each* $v_s \in V_s$ **do**
  $w = 2$, $ub = +\infty$, $lb = 0$, $\lambda_1 = \lambda_2 = \lambda_3 = 0$, $t = 0$;
  while $\frac{ub-lb}{ub} \geq \gamma_1$ **do**
    solve Eq. (19) with dynamic programming to get $\mathcal{L}_{\text{dual}}(\lambda_1, \lambda_2, \lambda_3)$ and $\{\tilde{K}_{v_r}\}$;
    if $\mathcal{L}_{dual}(\lambda_1, \lambda_2, \lambda_3) < ub$ **then**
      $ub = \mathcal{L}_{\text{dual}}(\lambda_1, \lambda_2, \lambda_3)$, $t = 0$;
    else
      if $t > t_{TH}$ **then**
        $w = w/2$, $t = 0$;
      else
        $t = t + 1$;
      end
    end
    get $\mathbf{S}$ and $\{K_{v_r}\}$ with $\{\tilde{K}_{v_r}\}$ and *Algorithm* 3;
    $lb = \mathbf{S}$, get $\{f(\lambda_1^n), f(\lambda_2^n), f(\lambda_3^n)\}$ with Eq. (20);
    get step-size μ^n with Eq. (21);
    get λ_1^{n+1}, λ_2^{n+1} and λ_3^{n+1} with Eq. (22);
    $\lambda_1 = \lambda_1^{n+1}$, $\lambda_2 = \lambda_2^{n+1}$, $\lambda_3 = \lambda_3^{n+1}$;
  end
end
return $\{K_{v_r}\}$;

D. Approximation Algorithm for Lower-level Problem

With a solution of the upper-level problem ($\{K_{v_r}\}$), we can formulate an MILP for the lower-level problem, as described in Section III-B. To solve the MILP time-efficiently, we propose a polynomial-time approximation algorithm based on linear programming (LP) relaxation and randomized rounding.

Algorithm 5 shows the procedure of the approximation algorithm. Regarding its inputs, M denotes the maximum number of rounding iterations, and γ_2 is the preset parameter for approximation. In *Line* 1, we preprocess the MILP for the lower-level problem and relax the result to get an LP. Here, the preprocessing is introduced to improve the efficiency of our problem-solving, and it includes two steps. Firstly, we remove Eq. (16) from the lower-level MILP, because it has noting to do with the optimization objective, and then the lower-level MILP becomes an ILP. Secondly, we add the following constraints to improve the successful rate of subsequent randomized rounding (*i.e.*, reducing the probability of obtaining infeasible solutions).

$$y_{(v_r,u_r)}^{v_s,u_s} \leq L_{v_s,u_s}^*, \quad \forall (v_r, u_r) \in E_r, \ \forall v_s, u_s \in V_s, \tag{24}$$

which will not affect the solution of the original lower-level MILP. In the LP relaxation, we relax all the binary variables in the ILP to real ones within $[0, 1]$.

The LP is solved in *Line* 2 to get $\{\tilde{P}_{v_r,v_s}^*\}$, $\{\tilde{L}_{v_s,u_s}^*\}$ and $\tilde{\mathbf{g}}$, which are all in real numbers. $\tilde{\mathbf{g}}$ is the objective of the LP, *i.e.*, the number of resource hot-spots, and we round it up in *Line* 3 and use it as the lower-bound of the optimal solution of the lower-level problem. Then, the subsequent while-loop builds a qualified approximation solution in iterations (*Lines* 4-32). In the while-loop, we use two for-loops to determine the values of $\{P_{v_r,v_s}^*\}$ and $\{L_{v_s,u_s}^*\}$ with randomized rounding, respectively (*Lines* 5-23). Specifically, we get the binary values of $\{P_{v_r,v_s}^*\}$ and $\{L_{v_s,u_s}^*\}$ randomly with $\{\tilde{P}_{v_r,v_s}^*\}$ and $\{\tilde{L}_{v_s,u_s}^*\}$ as the probabilities, respectively. Next, *Line* 24 checks whether the obtained $\{P_{v_r,v_s}^*\}$ and $\{L_{v_s,u_s}^*\}$ represents a feasible solution to the original lower-level MILP. If yes, we calculate the objective of the lower-level problem ($\mathbf{g}^*$) with $\{P_{v_r,v_s}^*\}$ and $\{L_{v_s,u_s}^*\}$ (*Line* 27). Otherwise, we proceed to the next iteration. Finally, when the objective $\mathbf{g}^*$ satisfies the approximation ratio, we break the while-loop and calculate the

Algorithm 5: LP Relaxation and Randomized Rounding to Solve Lower-level Problem

Input: $\{K_{v_r}\}$, M, γ_2
preprocess and relax the lower-level MILP with $\{K_{v_r}\}$ to get an LP;
solve the LP to get the values of $\{\tilde{P}_{v_r,v_s}^*\}$, $\{\tilde{L}_{v_s,u_s}^*\}$ and $\tilde{\mathbf{g}}$ in real numbers;
$\tilde{\mathbf{g}} = \lceil \tilde{\mathbf{g}} \rceil$, $n = 0$;
while $n < M$ **do**
  for *each* v_r *in* $\{v_r : K_{v_r} = 0\}$ **do**
    determine the SN that VM v_r should be migrated to randomly with $\{\tilde{P}_{v_r,v_s}^*\}$ as probabilities;
    update $\{P_{v_r,v_s}^*\}$ according to the result;
  end
  $\{L_{v_s,u_s}^* = 0\}$, $flag = 0$;
  for *each* $v_s \in V_s$ **do**
    for *each* $u_s \in V_s \setminus v_s$ **do**
      if $L_{v_s,u_s}^* = 1$ **then**
        $flag = 1$;
        break;
      end
    end
    if $flag = 1$ **then**
      continue;
    else
      get the SN u_s that v_s be connected to through OXC randomly with $\{\tilde{L}_{v_s,u_s}^*\}$ as probabilities;
      update $\{L_{v_s,u_s}^*\}$ according to the result;
    end
  end
  if $\{P_{v_r,v_s}^*\}$ *and* $\{L_{v_s,u_s}^*\}$ *do not denote a feasible solution to lower-level MILP* **then**
    continue;
  end
  calculate objective $\mathbf{g}^*$ with $\{P_{v_r,v_s}^*\}$ and $\{L_{v_s,u_s}^*\}$;
  if $\frac{\mathbf{g}^*}{\tilde{\mathbf{g}}} < 1 + \gamma_2$ **then**
    break;
  end
  $n = n + 1$;
end
calculate the latency of VM migration T with Eq. (16) using $\{P_{v_r,v_s}^*\}$ and $\{L_{v_s,u_s}^*\}$;
return $\{P_{v_r,v_s}^*\}$, $\{L_{v_s,u_s}^*\}$, and T;

latency of VM migration with the obtained solution.

We can verify that the approximation ratio of *Algorithm* 5 is at most $1+\gamma_2$ as follows. Because the lower-level MILP is for minimization, the LP's solution (*i.e.*, $\tilde{\mathbf{g}}$) provides a lower-bound on the optimal solution, while the feasible solution built by *Algorithm* 5 (*i.e.*, $\mathbf{g}^*$) is an upper-bound. Hence, if we denote the optimal solution as $\mathbf{g}_{\text{MILP}}$, the approximation ratio of *Algorithm* 5 can be calculated as

$$\epsilon_2 = \frac{\mathbf{g}^*}{\mathbf{g}_{\text{MILP}}} \leq \frac{\mathbf{g}^*}{\tilde{\mathbf{g}}} < 1+\gamma_2. \tag{25}$$

We also would like to point out that according to the principle of LP relaxation with randomized rounding and the well-known Chernoff-Bound [48], the probability of *Algorithm* 5 obtaining a qualified solution can approach to 1, as long as the values of M and γ_2 are properly selected. Finally, as the LP solving in *Line* 2 can also be finished in polynomial time, *Algorithm* 5 is a polynomial-time approximation algorithm.

V. Performance Evaluation

In this section, we perform extensive simulations and discuss the results to evaluate the performance of our proposal.

A. Simulation Setup

As the classic fat-tree in Fig. 1(a) is one of the most popular topologies for EPS-based inter-rack networks, the simulations assume that a classic k-ray fat-tree is used for the EPS part of the HOE-DCN (*e.g.*, the fat-tree in Fig. 1(a) has $k=4$). Meanwhile, we assume that the oversubscription ratio in the EPS-based inter-rack network is $\frac{C_{\text{BW}}}{B_{\text{E}}} = 2$ for each rack. It should be noted that considering the numerous racks in a DCN and the prohibitive complexity of managing all of them as a whole, an operator usually manages its DCN in a modular way. Specifically, the operator can divide its DCN into many points-of-delivery (PoDs), and treat each PoD as a module of network, compute and storage components that work together to deliver network services [49]. Hence, we also divide each of the HOE-DCNs considered in the simulations into PoDs, and assume that each VNT can only be mapped and remapped within one PoD. Specifically, for an HOE-DCN that uses the k-ray fat-tree as the EPS part, we divide it into $\frac{k}{2}$ PoDs and each PoD includes k racks. Regarding the size of the HOE-DCN, we surveyed the commonly-used scales for fat-trees, and decide to architect the largest HOE-DCN in our simulations based on the 128-ray fat-tree [50].

As for the benchmark, we design a weight-based single-level optimization, also to balance the tradeoff between the two objectives considered in our proposed bilevel model.

$$\begin{aligned} \text{Minimize} \quad & \alpha \cdot T + \sum_{v_s \in V_s} \left(g_{v_s}^{\text{IT}} + g_{v_s}^{\text{EO}} + g_{v_s}^{\text{EI}} \right). \\ \text{s.t.} \quad & \text{Eqs. (6)} - \text{(16)}, \end{aligned} \tag{26}$$

where α is the weight for minimizing the estimated latency of VM migration (*i.e.*, T). Here, we define $\alpha = 3 \cdot |V_s|$ to ensure that minimizing T is still the primary objective as that in our bilevel model. We solve the optimization in Eq. (26) with LP relaxation and randomized rounding, with the procedure that is similar to that in *Algorithm* 5.

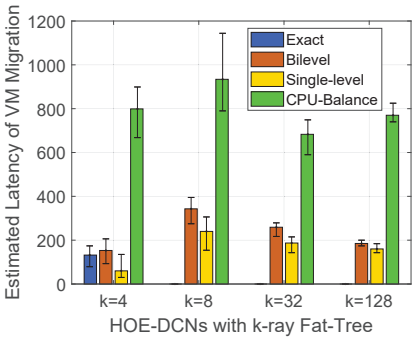
To make sure that the results are generic, we use "units" to denote the units of bandwidth and IT resources and "time-units" to represent the units of time and latency. Note that, although their units are not the physical ones, we do choose the distributions of the parameters according to the cases in real-world DCNs [51] or based on our own observations in experiments (*e.g.*, in [7, 8]), for ensuring the practicalness of our simulations. For the k-ray fat-tree, the CPU, memory and I/O resource capacities of each rack is assume to be $C_{\text{CPU}} = 100 \cdot k$ units and $C_{\text{MEM}} = 128 \cdot k$ units and $C_{\text{BW}} = 1000 \cdot k$ units/time-unit, respectively, the bandwidth capacity of each ToR switch to/from the EPS part is set as $B_{\text{E}} = 1000 \cdot \frac{k}{2}$ units/time-unit, and the available bandwidth on the ToR switch in SN v_s to/from the EPS part (*i.e.*, $B_{v_s}^{\text{EO}}/B_{v_s}^{\text{EI}}$) is randomly selected within $[500 \cdot \frac{k}{2}, 1000 \cdot \frac{k}{2}]$ units/time-unit.

The simulations consider four HOE-DCNs, which are based on $\{4, 8, 32, 128\}$-ray fat-trees. Each VNT includes $[2, 20]$ VMs whose connectivity is set as 0.3. We include two types of VNTs in the simulations, *i.e.*, the IT-bound and I/O-bound ones. The CPU resource demand of each VM in a IT-bound or I/O-bound VNT is selected within $[1, 64]$ units or $[1, 12]$ units, respectively. The memory demand of each VM in a IT-bound or I/O-bound VNT is selected with in $[1, 128]$ units or $[1, 96]$ units, respectively. The bandwidth demand of each VL in a IT-bound or I/O-bound VNT is chosen within $[1, 1000]$ units/time-unit or $[1, 2500]$ units/time-unit, respectively. The size of data transfers to migrate a VM is set within $[10000, 40000]$ units. The thresholds for identifying resource hot-spots (η_{IT} and η_{BW}) are both set as 70%, and the N in *Algorithm* 2 is chosen as 7.

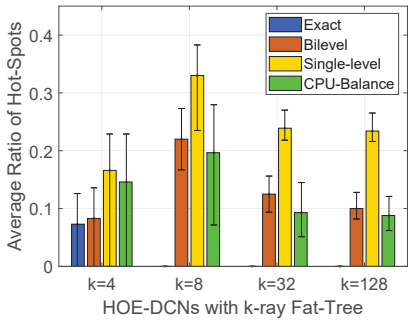
Our simulations run in the discrete-time way as follows. At the beginning of each simulation, we embed each VNT in a PoD of the HOE-DCN according to its resource demands, with the global resource capacity based VNE algorithm in [52], such that the IT and bandwidth resource usages in the HOE-DCN are balanced at the time of initial VNT embedding. Then, the resource demands of each VNT change over time randomly according to the aforementioned ranges. At each maintenance time, we gather the future demands of each VNT to determine whether the IT and bandwidth resources on racks will be used up. If yes, VNT remapping will be triggered. This is repeated until the total simulation time expires. To maintain sufficient statistical accuracy, the simulations average the results from 60 independent runs to get each data point. Meanwhile, to show the stability of the algorithms in the simulations, we also mark the range of the 95% confidence interval in Figs. 3 and 4.

B. Performance in HOE-DCNs in Different Scales

We first evaluate the algorithms with HOE-DCNs in different scales. Fig. 3 shows the simulation results, where the ratio between IT-bound and I/O-bound VNTs is set as $1:1$. Here, "Exact" refers to *Algorithm* 1, which can obtain the exact solution of our bilevel model, "Bilevel" denotes our proposed approximation algorithm (*Algorithm* 2), "Single-level" is the single-level benchmark mentioned in the previous subsection, and "CPU-Balance" refers to the VNT remapping algorithm developed in [24], which tries to remap VNTs to



(a) Estimated latency of VM migration



(b) Average ratio of resource hot-spots

Fig. 3. Results for HOE-DCNs in different scales (Exact only runs in the 4-ray fat-tree).

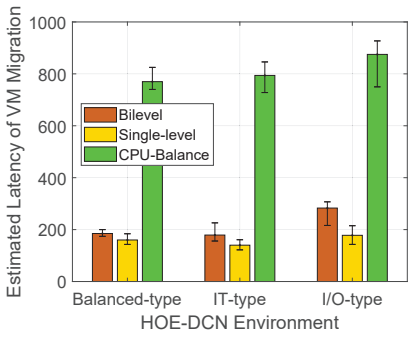
balance the CPU usages on the racks in an HOE-DCN. We set $\gamma_1 = \gamma_2 = 0.1$ for Bilevel to ensure that its approximation ratio is at least 0.9, and similarly, the approximation ratios of Single-level and CPU-Balance are also maintained above 0.9. Due to the time complexity of Exact, it can only be solved for the HOE-DCN with 4-ray fat-tree. Fig. 3(a) shows the results on estimated latency of VM migration, while the results in Fig. 3(b) denote the average ratio of the number of resource hot-spots to its upper-limit (*i.e.*, $\frac{\mathbf{g}^*}{\hat{\mathbf{g}}}$). For the HOE-DCN with 4-ray fat-tree, the results in Fig. 3 confirm that the solution of Bilevel is close to that of Exact, for both the estimated latency of VM migration and the average ratio of resource hot-spots.

As the primary objective of Single-level is to minimize the estimated latency of VM migration, it performs the best among the four algorithms in Fig. 3(a). However, it also provides much larger average ratio of resource hot-spots in Fig. 3(b) than Exact, Bilevel and CPU-Balance, which suggests that the single-level optimization has difficulty to balance the tradeoff between the primary and secondary objectives well with empirically-assigned weights. Similarly, as the objective of CPU-Balance is to balance the CPU usages on the racks, it can reduce the number of resource hot-spots in Fig. 3(b), with a relatively large number of VM migrations. Hence, its estimated latency of VM migration is the longest in Fig. 3(a). On the other hand, Bilevel balances the tradeoff much better than Single-level and CPU-Balance, especially for the large-scale HOE-DCNs (*i.e.*, those with $\{32, 128\}$-ray fat-trees).

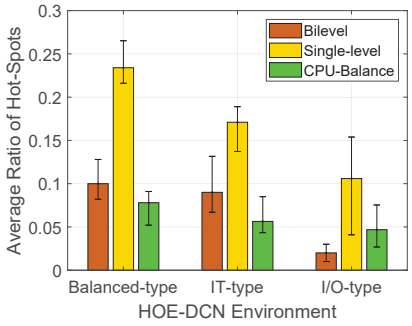
Table V shows the running time of the algorithms. It can be seen that Exact can only be solved for the HOE-DCN in the smallest scale, while Bilevel, Single-level and CPU-Balance are much more time-efficient. The running time of Bilevel is the shortest, which is significantly shorter than that of Single-level. This is because for Bilevel and Single-level, the procedure of LP relaxation and randomized rounding takes the major part of their running time, while Single-level has more constraints to handle in the procedure.

TABLE V
RUNNING TIME OF ALGORITHMS (SECONDS)

Scale of HOE-DCN	$k = 4$	$k = 8$	$k = 32$	$k = 128$
Exact	1438.854	-	-	-
Bilevel	0.0001	0.008	0.434	13.258
Single-level	0.0001	0.010	0.634	35.214
CPU-Balance	0.0001	0.010	0.444	14.137



(a) Estimated latency of VM migration

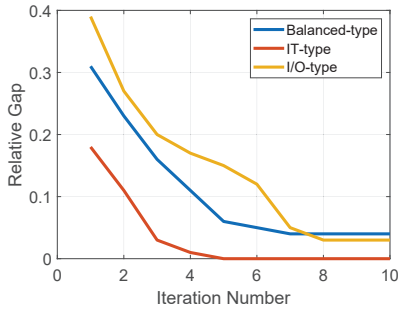


(b) Average ratio of resource hot-spots

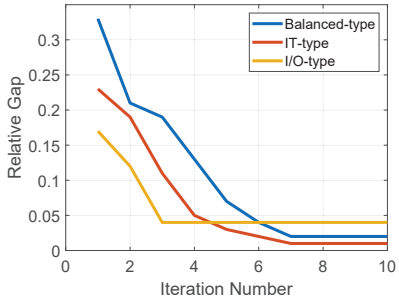
Fig. 4. Results for HOE-DCNs with different VNTs (128-ray fat-tree).

C. Performance in HOE-DCNs with Different VNTs

Note that, the network services in DCNs can use different types of VNTs, *e.g.*, the VNTs focus their resource usages on IT and bandwidth resources can be classified as IT-bound and I/O-bound ones, respectively [7]. For instance, a network service of MapReduce [14] usually organizes its VMs (*i.e.*, the name-nodes and data-nodes) as a cluster-type VNT. Here, a typical MapReduce service such as WordCount (*i.e.*, counting word occurrences in a give set) usually focuses its resource usages on CPU cycles and memory, and thus it uses an IT-bound VNT, while another typical service of MapReduce, namely, Teragen, which generates and distributes random data, consumes much more bandwidth than WordCount and hence can be classified as I/O-bound [17]. Meanwhile, there are also VNTs that have alternate bandwidth-intensive and computing-intensive phases, *e.g.*, those for distributed machine learning



(a) Convergence of *Algorithm* 4



(b) Convergence of *Algorithm* 5

Fig. 5. Convergence performance of Bilevel (128-ray fat-tree).

[53]. To address these practical cases, we consider three types of HOE-DCN scenarios, where the ratio between IT-bound and I/O-bound VNTs is set as $1:1$, $4:1$ and $1:4$, namely, Balanced-type, IT-type and I/O-type, respectively.

This time, we only consider the HOE-DCN with 128-ray fat-tree. The results in Fig. 4 still illustrate that Bilevel can balance the two objectives much better than the benchmarks. Meanwhile, it is interesting to notice that the average ratios of resource hot-spots obtained in the I/O-type HOE-DCN environment are much smaller than those obtained in the other two environments. This is due to the abundant bandwidth provided by the OCS part, *i.e.*, if the resource hot-spots are mainly caused by high bandwidth usages on VLs, they can be easily addressed with the VNT remapping that reconfigures the VLs onto optical connections. For these large-scale problems, we also analyze the convergence performance of Bilevel, and plot the convergence of the approximation algorithms for the upper and lower optimizations (*i.e.*, *Algorithms* 4 and 5, respectively) in Fig. 5. We observe that the relative gaps of both algorithms can be reduced to below 0.1 in only a few iterations, which further verifies the time-efficiency of Bilevel.

Finally, we would like to confirm that for the large-scale problems, Bilevel can always balance the tradeoff between the two objectives better than the benchmarks, no matter what weight α is used in Eq. (26). Fig. 6 shows the results, where we plot the average ratio of resource hot-spots *versus* the estimated latency of VM migration, to show the tradeoff clearly. Each curve of Single-level is obtained by changing the value of α. It can be seen that in each case, the data point for the results from Bilevel is always below the curve for the results from Single-level. This verifies that Bilevel balances the tradeoff better than Single-level, regardless of the choice of α, *i.e.*, the hassle of empirical parameter adjustments can

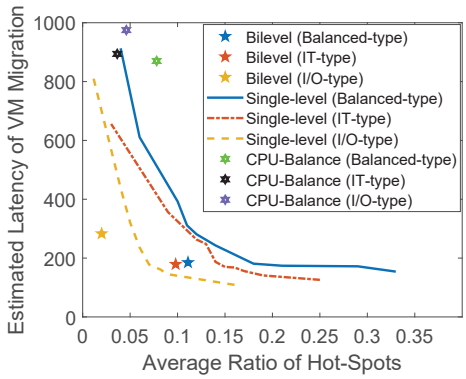


Fig. 6. Performance on tradeoff balancing (128-ray fat-tree).

be avoided by Bilevel. Meanwhile, Fig. 6 also indicates that Bilevel balances the tradeoff much better than CPU-Balance.

VI. Conclusion

In this work, we studied the problem of VNT remapping in an HOE-DCN from a novel perspective, *i.e.*, the remapping schemes should be optimized for not only the network status after the remapping but also the transition to realize it. Specifically, we modeled the problem of VNT remapping as a bilevel optimization, where the upper-level optimization aims at selecting proper VMs to migrate such that the estimated latency of VM migration can be minimized, and the lower-level optimization determines the actual scheme of VNT remapping for minimizing the number of resource hot-spots. We first formulated a BMILP model for the bilevel optimization, and then proposed a polynomial time algorithm based on enumeration to solve it directly but approximately. Extensive simulations confirmed the effectiveness of our proposal, and the results verified that it can get near-optimal solutions whose performance gaps to the optimal ones are bounded, and balance the tradeoff between the two objectives much better than the benchmarks based on single-level optimizations.

Acknowledgments

This work was supported in part by the NSFC project 61871357, SPR Program of CAS (XDC02070300), Fundamental Funds for Central Universities (WK3500000006), and the Science Foundation of Shenzhen City under grants JSGG20191127151401743 and JCYJ20190808165401679.

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] Cisco Visual Networking Index, 2017-2022. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html.

[3] Y. Tian, R. Dey, Y. Liu, and K. Ross, "Topology mapping and geolocating for China's Internet," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, pp. 1908–1917, Sept. 2012.

[4] W. Lu *et al.*, "AI-assisted knowledge-defined network orchestration for energy-efficient data center networks," *IEEE Commun. Mag.*, vol. 58, pp. 86–92, Jan. 2020.

[5] N. Farrington *et al.*, "Helios: a hybrid electrical/optical switch architecture for modular data centers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 339–350, Oct. 2010.

[6] K. Chen *et al.*, "OSA: An optical switching architecture for data center networks with unprecedented flexibility," *IEEE/ACM Trans. Netw.*, vol. 22, pp. 498–511, Apr. 2013.

[7] H. Fang *et al.*, "Predictive analytics based knowledge-defined orchestration in a hybrid optical/electrical datacenter network testbed," *J. Lightw. Technol.*, vol. 37, pp. 4921–4934, Oct. 2019.

[8] Q. Li *et al.*, "Scalable knowledge-defined orchestration for hybrid optical/electrical datacenter networks," *J. Opt. Commun. Netw.*, vol. 12, pp. A113–A122, Feb. 2020.

[9] Z. Zhu, W. Lu, L. Zhang, and N. Ansari, "Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing," *J. Lightw. Technol.*, vol. 31, pp. 15–22, Jan. 2013.

[10] L. Gong *et al.*, "Efficient resource allocation for all-optical multicasting over spectrum-sliced elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. 836–847, Aug. 2013.

[11] Y. Yin *et al.*, "Spectral and spatial 2D fragmentation-aware routing and spectrum assignment algorithms in elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. A100–A106, Oct. 2013.

[12] N. Bitar, S. Gringeri, and T. Xia, "Technologies and protocols for data center and cloud networking," *IEEE Commun. Mag.*, vol. 51, pp. 24–31, Sept. 2013.

[13] M. Bari *et al.*, "Data center network virtualization: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, pp. 909–928, Second Quarter 2013.

[14] D. Borthakur, *The Hadoop Distributed File System: Architecture and Design*. Apache Software Foundation, 2007.

[15] L. Gong and Z. Zhu, "Virtual optical network embedding (VONE) over elastic optical networks," *J. Lightw. Technol.*, vol. 32, pp. 450–460, Feb. 2014.

[16] 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.

[17] B. Kong *et al.*, "Demonstration of application-driven network slicing and orchestration in optical/packet domains: On-demand vDC expansion for Hadoop MapReduce optimization," *Opt. Express*, vol. 26, pp. 14 066–14 085, May 2018.

[18] J. Duan and Y. Yang, "A load balancing and multi-tenancy oriented data center virtualization framework," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, pp. 2131–2144, Aug. 2017.

[19] J. Liu *et al.*, "On dynamic service function chain deployment and readjustment," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, pp. 543–553, Sept. 2017.

[20] S. Zhao, D. Li, K. Han, and Z. Zhu, "Proactive and hitless vSDN reconfiguration to balance substrate TCAM utilization: From algorithm design to system prototype," *IEEE Trans. Netw. Serv. Manag.*, vol. 16, pp. 647–660, Jun. 2019.

[21] C. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," *IEEE Trans. Comput.*, vol. C-34, pp. 892–901, Oct. 1985.

[22] Polatis Series 7000 Software-Defined Optical Circuit Switch. [Online]. Available: https://www.polatis.com/series-7000-384x384-port/-software-controlled-optical-circuit-switch-sdn-enabled.asp

[23] S. Zhao and Z. Zhu, "Network service reconfiguration in hybrid optical/electrical datacenter networks," in *Proc. of ONDM 2020*, pp. 1–6, May 2020.

[24] S. Zhao and Z. Zhu, "On virtual network reconfiguration in hybrid optical/electrical datacenter networks," *J. Lightw. Technol.*, vol. 38, pp. 6424–6436, Aug. 2020.

[25] A. Toutov, A. Vorozhtsov, and N. Toutova, "Estimation of total migration time of virtual machines in cloud data centers," in *Proc. of IT&QM&IS 2018*, pp. 389–393, Nov. 2018.

[26] A. Choudhary *et al.*, "A critical survey of live virtual machine migration techniques," *J. Cloud Comp.*, vol. 6, pp. 23:1–41, Nov. 2017.

[27] J. Luo, X. Fan, and L. Yin, "Communication-aware and energy saving virtual machine allocation algorithm in data center," in *Proc. of HPCC/SmartCity/DSS 2019*, pp. 819–826, Oct. 2019.

[28] Q. Lv, F. Zhou, and Z. Zhu, "On the bilevel optimization to design control plane for SDONs in consideration of planned physical-layer attacks," *IEEE Trans. Netw. Serv. Manag.*, vol. 37, pp. 1113–1122, Feb. 2021.

[29] 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.

[30] M. Rabbani *et al.*, "On tackling virtual data center embedding problem," in *Proc. of IFIP/IEEE IM 2013*, pp. 177–184, May 2013.

[31] X. Wen *et al.*, "Towards reliable virtual data center embedding in software defined networking," in *Proc. of MILCOM 2016*, pp. 1059–1064, Nov. 2016.

[32] W. Fang *et al.*, "Joint defragmentation of optical spectrum and IT resources in elastic optical datacenter interconnections," *J. Opt. Commun. Netw.*, vol. 7, pp. 314–324, Mar. 2015.

[33] R. Munoz *et al.*, "Integrated SDN/NFV management and orchestration architecture for dynamic deployment of virtual SDN control instances for virtual tenant networks [invited]," *J. Opt. Commun. Netw.*, vol. 7, pp. B62–B70, Nov. 2015.

[34] J. Yin *et al.*, "Experimental demonstration of building and operating QoS-aware survivable vSD-EONs with transparent resiliency," *Opt. Express*, vol. 25, pp. 15 468–15 480, 2017.

[35] Z. Zhu *et al.*, "Build to tenants' requirements: On-demand application-driven vSD-EON slicing," *J. Opt. Commun. Netw.*, vol. 10, pp. A206–A215, Feb. 2018.

[36] Y. Cui *et al.*, "Traffic-aware virtual machine migration in topology-adaptive DCN," *IEEE/ACM Trans. Netw.*, vol. 25, pp. 3427–3440, Sept. 2017.

[37] H. Wang, Y. Li, Y. Zhang, and D. Jin, "Virtual machine migration planning in software-defined networks," *IEEE Trans. Cloud Comput.*, vol. 7, pp. 1168–1182, May 2019.

[38] S. Zhao, X. Pan, and Z. Zhu, "On the parallel reconfiguration of virtual networks in hybrid optical/electrical datacenter networks," in *Proc. of GLOBECOM 2020*, pp. 1–6, Dec. 2020.

[39] M. Najm and V. Tamarapalli, "VM migration for profit maximization in federated cloud data centers," in *Proc. of COMSNETS 2020*, pp. 882–884, Mar. 2020.

[40] A. Zhou, S. Wang, X. Ma, and S. Yau, "Towards service composition aware virtual machine migration approach in the cloud," *IEEE Trans. Serv. Comput.*, vol. 13, pp. 735–744, Dec. 2020.

[41] J. Bard, "Some properties of the bilevel programming problem," *J. Optimiz. Theory App.*, vol. 68, pp. 371–378, Feb. 1991.

[42] P. Poirion, S. Toubaline, C. D'Ambrosio, and L. Liberti, "Bilevel mixed-integer linear programs and the zero forcing set," *Optimiz.*, pp. 1–15, Nov. 2015. [Online]. Available: http://www.optimization-online.org/DB_FILE/2015/11/5210.pdf.

[43] H. Haghighat and B. Zeng, "Bilevel mixed integer transmission expansion planning," *IEEE Trans. Power Syst.*, vol. 33, pp. 7309–7312, Aug. 2018.

[44] B. Zeng and Y. An, "Solving bilevel mixed integer program by reformulations and decomposition," *Optimiz.*, pp. 1–34, Jun. 2014. [Online]. Available: http://www.optimization-online.org/DB_FILE/2014/07/4455.pdf.

[45] J. Moore and J. Bard, "The mixed integer linear bileve programming problem," *Oper. Res.*, vol. 38, pp. 911–921, Jul. 1990.

[46] M. Held, P. Wolfe, and H. Crowder, "Validation of subgradient optimization," *Math. Program.*, vol. 6, pp. 62–88, Dec. 1974.

[47] D. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999.

[48] D. Dubhashi and A. Panconesi, *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[49] Point of delivery (networking). [Online]. Available: https://en.wikipedia.org/wiki/Point_of_delivery_(networking)#cite_note-1.

[50] S. Zafar, A. Bashir, and S. Chaudhry, "On implementation of DCTCP on three-tier and fat-tree data center network topologies," *SpringerPlus*, vol. 5, pp. 1–18, Jun. 2016.

[51] Amazon EC2 instances. [Online]. Available: https://aws.amazon.com/cn/ec2/instance-types/.

[52] 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.

[53] M. Li *et al.*, "Scaling distributed machine learning with the parameter server," in *Proc. of OSDI 2014*, pp. 583–598, Oct. 2014.