On Virtual Network Reconfiguration in Hybrid Optical/Electrical Datacenter Networks

Sicheng Zhao and Zuqing Zhu, *Senior Member, IEEE*

***Abstract*—Hybrid optical/electrical datacenter networks (HOE-DCNs) build inter-rack networks with both electrical Ethernet switches and optical cross-connects (OXCs), and have been considered as a promising DCN architecture. However, to adapt to the dynamic network environment, the reconfiguration of virtual networks (VNTs) in an HOE-DCN still faces the unique difficulty that the HOE-DCN's topology can change because of the one-to-one connectivity of OXCs. To the best of our knowledge, this problem still has not been fully explored. In this paper, we address this problem, and consider how to achieve effective VNT reconfiguration in an HOE-DCN such that the IT resource usages in racks can be re-balanced with the migration of virtual machines (VMs). We first formulate a mixed integer linear programming (MILP) to describe the VNT reconfiguration. Then, we solve the problem with two steps, 1) obtaining the VM migration schemes to balance the loads on racks, and 2) determining the reconfiguration schemes of related virtual links (VLs) and the OXC. For the first step, we propose a polynomial-time approximation algorithm by leveraging linear relaxation. Then, we tackle the optimization of the second step by developing an algorithm that involves a linear-time dynamic programming and an integer linear programming (ILP). To solve the ILP time-efficiently, we propose another polynomial-time approximation algorithm based on Lagrangian relaxation. Our simulations confirm the effectiveness of the proposed approximation algorithms, and verify that the overall procedure including them outperforms the existing approach.**

***Index Terms*—Hybrid optical/electrical datacenter network (HOE-DCN), Network virtualization, Virtual network reconfiguration, VM migration, Approximation algorithm.**

I. Introduction

NOWADAYS, datacenters (DCs) have already become the biggest contributor to Internet traffic, and the traffic in DCs has being increasing rapidly with an annual rate close to 30% [1, 2]. Hence, considering the fast development of data-intensive network services such as Big Data and video streaming [3–5], we can estimate that the infrastructure of DC networks (DCNs) will face increasing challenges from architecture scalability, energy efficiency, and management agility [6], due to the pressure from enormous amounts of traffic. To address these challenges, people have proposed to add optical circuit switching (OCS) into inter-rack networks and integrate it with the conventional electrical packet switching (EPS) [7, 8]. By doing so, one realizes a hybrid optical/electrical DCN (HOE-DCN), which can be more scalable and energy-efficient [9, 10]. Compared with EPS, OCS provides larger switching capacity and higher energy efficiency, while its downside is longer setup and reconfiguration latency [11–16]. Therefore, it would be interesting to study how to operate HOE-DCNs in consideration of the pros and cons of EPS and OCS [17].

S. Zhao and Z. Zhu are with the School of Information Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, P. R. China (email: zqzhu@ieee.org).

Manuscript received on March 31, 2020.

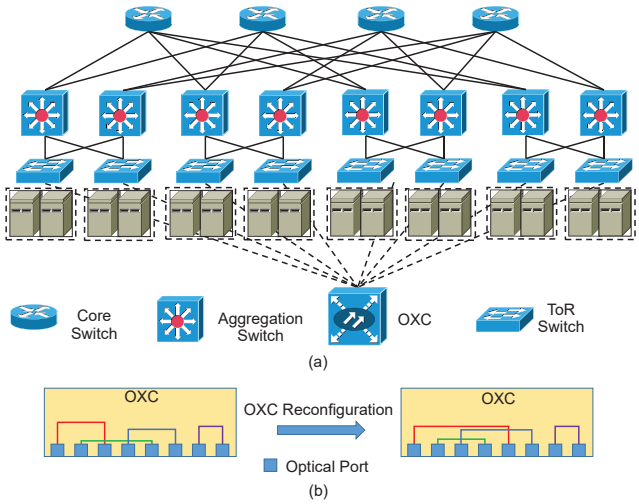


Fig. 1. (a) Architecture of HOE-DCN, and (b) Reconfiguration of OXC.

The typical network architecture of an HOE-DCN can be seen in Fig. 1(a), where the top-of-rack (ToR) switches are interconnected with two types of inter-rack networks. In the figure, the EPS-based one is on the top, which consists of electrical Ethernet switches organized in a hieratical topology, while the optical cross-connect (OXC) at the bottom represents the OCS-based inter-rack network. Therefore, the DCN operator can route traffic flows over the two inter-rack networks, according to their characteristics. However, the network control and management (NC&M) for a DCN is actually much more complicated than traffic routing. This is because a network service handled by a DCN normally deploys multiple virtual machines (VMs), and relies on the VMs' collaboration to accomplish service tasks. For example, Hadoop MapReduce [18] usually relies on VM clusters, each of which includes both name- and data-nodes, to run its tasks.

Hence, each network service actually forms a virtual network (VNT) [9], where the VMs are the virtual nodes (VNs) and the routing paths to bridge the communications among the VMs are the virtual links (VLs). This motivates us to leverage the well-known virtual network embedding (VNE) [19, 20] for deploying network services in a DCN, *i.e.*, the DCN is treated as the substrate network (SNT) shared by the VNTs for network services. However, VNE is just for the initial deployment, while each network service needs to be maintained throughout its lifetime. This is because the

network environment of a DCN is usually highly dynamic, *i.e.*, the usages of IT and bandwidth resources are time-variant, and network services can arrive, change and leave on-the-fly [21]. Therefore, the optimality of initial VNE results can be progressively degraded over time. This suggests that VNT reconfiguration [22–24] has to be considered to re-optimize the resource allocation in the DCN from time to time. However, VNT reconfiguration is more complex than VNE because it needs to select the VNTs to reconfigure, which does not exist in VNE. Moreover, to limit the operational complexity of VNT reconfiguration, we should only reconfigure a restricted number of VMs and VLs, which brings in additional constraints.

Note that, even though VNT reconfiguration in generic packet networks [24] and DCNs [22] has already been investigated, the VNT reconfiguration in HOE-DCNs is a brand-new problem that is intrinsically more complex. This is because all the existing studies on this topic [22–24] are based on the assumption that the SNT's topology will not change after each reconfiguration operation, while this assumption is not valid in HOE-DCNs. As shown in Fig. 1(b), the operation principle of OXCs determines that they can only achieve one-to-one connectivity between inputs and outputs. Hence, if the VNT reconfiguration in an HOE-DCN wants to remap one or more VLs on an optical connection through the OXC, the SNT's topology might be changed. In other words, after the OXC has been reconfigured, the OCS-based inter-rack network will have different physical connections among the ToR switches, which will not happen in a conventional DCN.

Previously, in [25], we have conducted a preliminary study on how to realize effective VNT reconfiguration in an HOE-DCN, such that the IT resource utilizations in racks can be re-balanced with VM migration. The network model was based on our implementations of network orchestration systems for HOE-DCNs in [9, 10], and thus practical assumptions were used to ensure that algorithms developed based on them can be deployed in a real-world HOE-DCN without any difficulty. First of all, we selected the VMs, which are running on heavy-loaded racks and thus should be migrated, with a trivial greedy algorithm based on empirically-determined parameters. Then, we designed an algorithm to calculate new VNE schemes of the VNTs, which have VMs that have been selected for migration. More specifically, the problem solving was divided into two steps, 1) calculating VM migration schemes for the VNTs to balance the loads on racks, and 2) determining reconfiguration schemes of the related VLs and OXC, and time-efficient heuristics were designed to tackle them. However, the heuristics developed in [25] cannot get near-optimal solutions whose performance gaps to the optimal ones are bounded.

The aforementioned dilemma motivates us to extend the study in this work. Specifically, for the problem of VNT reconfiguration in HOE-DCNs, we still focus on the algorithm design to obtain new VNE schemes of the VNTs that have VMs to migrate, but design approximation algorithms for the two steps mentioned above. The new contributions made in this work are explained as follows. Firstly, we formulate a mixed integer linear programming (MILP) model to describe the overall optimization for calculating new VNE schemes based on preselected VMs. Secondly, to determine where to migrate the selected VMs such that the loads on the racks can be re-balanced, we propose a polynomial-time approximation algorithm by leveraging linear relaxation. Thirdly, once the racks that the selected VMs migrate to are determined, we solve the subproblem of calculating the reconfiguration schemes of related VLs and the OXC with an algorithm that involves a linear-time dynamic programming and an integer linear programming (ILP) model. In order to solve the ILP time-efficiently, we propose another polynomial-time approximation algorithm based on Lagrangian relaxation. Therefore, the study in this work greatly improves the algorithm designs in [25], because it ensures optimization gaps of the obtained solutions. Performance evaluations with extensive simulations confirm the effectiveness of the proposed approximation algorithms.

The rest of the paper is organized as follows. We survey the related work briefly in Section II. Section III provides the problem description. The overall optimization model of VNT reconfiguration in an HOE-DCN is presented in Section IV. In Section V, we propose the approximation algorithms, and the simulations for performance evaluations are discussed in Section VI. Finally, Section VII summarizes the paper.

II. Related Work

Since the inception of network virtualization, the problem of VNE has been studied intensively for various types of networks [20, 26–31]. Specifically, the studies in [30, 31] have addressed the VNE in DCNs, which leveraged the IT and bandwidth resources on servers and network links, respectively, to deploy the VMs and VLs of VNTs. One can refer to [32] for a comprehensive survey on the existing VNE algorithms. Meanwhile, the network virtualization technologies have been reviewed in [33]. To address the dynamic nature of DCNs, network reconfiguration schemes should be considered to re-balance the usages of IT and bandwidth resources frequently [34]. Note that, in DCNs, re-balancing resource usages, especially the IT resource usages, is a commonly-used mechanism to avoid overloaded hot-spots (*i.e.*, resource contentions) [35], and it can bring in a few benefits, such as reducing job completion time [36], and saving power consumption [37].

Without considering HOE-DCNs as SNTs, the studies in [22, 24] studied how to realize VNT reconfiguration. The authors of [22] considered how to leverage VNT reconfiguration to realize load balancing in a conventional DCN built with EPS-based Ethernet switches. In [24], we studied the problem of reconfiguring virtual software-defined networks (vSDNs) to balance the flow-table installations in an SNT that consists of programmable data plane switches. On the other hand, the network virtualization systems, which can realize VNT reconfiguration, have been experimentally demonstrated in [38–40] for load balancing and addressing physical-layer issues. As none of the existing studies on VNT reconfiguration used an HOE-DCN as the SNT, they all assumed that the SNT's topology will not change through the reconfiguration. This, however, is invalid for the problem considered in this work, because in an HOE-DCN, the reconfiguration of OXCs results in different physical connections among the ToR switches.

The architecture of HOE-DCN has been proposed in [7, 8] to integrate the advantages of EPS and OCS for making

DCNs more scalable and energy-efficient. Recently, the studies in [6, 9, 10] suggested that by utilizing artificial intelligence technologies such as deep reinforcement learning (DRL), one can effectively improve the management agility of HOE-DCNs and stimulate EPS and OCS to cooperate seamlessly for application-aware service provisioning. Nevertheless, these studies were focused on the DRL-assisted network orchestration and related experimental demonstrations, but did not address how to solve the VNT reconfiguration in an HOE-DCN. As we have explained before, the major difficulty of solving VNT reconfiguration for HOE-DCNs is the unique operation principle of OXCs, which restricts one-to-one connectivity for ToR switches. This implies that the VNT reconfiguration in an HOE-DCN can change the SNT's topology from time to time. To address this new problem, we conducted a preliminary study in [25], and developed several heuristics that cannot ensure bounded performance gaps to optimal solutions. However, as the heuristics do not have performance guarantees, the problem is still not fully explored.

III. Problem Description

In this section, we explain the network model, procedure, and preprocessing of VNT reconfiguration in HOE-DCNs.

A. Network Model

We model the SNT (*i.e.*, an HOE-DCN) as $G(V_s, E_s)$, where V_s and E_s are the sets of substrate nodes (SNs) and substrate links (SLs), respectively. In our problem, each SN $v_s \in V_s$ is a server rack, which includes a ToR switch, and a server pool whose total IT and I/O capacities are denoted as C_{v_s} and B_{v_s}, respectively. Before VNT reconfiguration, the IT and I/O resource utilizations on rack v_s are c_{v_s} and b_{v_s}, respectively. For each rack, its ToR switch has SLs connecting to the OXC and the EPS-based inter-rack network simultaneously as shown in Fig. 2. Hence, an SL $e_s \in E_s$ can be either an Ethernet link for EPS or an optical connection for OCS (*i.e.*, to/from the OXC). At any given time, due to the one-to-one connectivity of the OXC, each ToR switch can only communicate with one other ToR switch using OCS, while who to talk with is determined by the OXC's configuration.

We model the topology of a VNT as $G_r(V_r, E_r)$, where V_r and E_r are the sets of VNs and VLs, respectively. Here, each VN $v_r \in V_r$ represents a VM that runs the network service of the VNT, and its IT resource demand is denoted as c_{v_r}. A VL $e_r = (v_r, u_r) \in E_r$ connects two VMs (v_r and u_r), and has a bandwidth requirement of $b_{(v_r,u_r)}$.

In this work, we assume that the EPS-based inter-rack network is architected based on a non-blocking topology, *e.g.*, the well-known fat-tree topology in Fig. 1(a). Hence, the bandwidth capacity between any two racks in the HOE-DCN will be enough to route all the flows between the servers in them, provided that there is no congestion on the intra-rack links between the servers and their ToR switches. We denote the I/O resource demand of a VM v_r as b_{v_r}, which equals the total bandwidth demand of all the VLs that end at it. In addition, we assume that each VL can be either "optical-preferred" or "do-not-care". Note that, this attribute is predetermined based on the traffic condition on the VL [25], and will not change afterwards. For instance, in a VNT for Hadoop applications, the traffic volume between two data-node VMs is much higher than that between data-node and name-node VMs [9, 10]. Hence, a VL between two data-node VMs should be predetermined as an "optical-preferred" one.

B. VNT Reconfiguration in HOE-DCNs

The overall procedure of the VNT reconfiguration in an HOE-DCN is explained with *Algorithm* 1 [25]. The basic idea of our VNT reconfiguration is to balance the IT resource utilizations in racks with VM migration. Note that, the VNT reconfiguration does not try to balance the bandwidth usages on SLs. The rationale behind this consideration is two-fold. Firstly, in DCNs, compared with the IT resource usages in racks, bandwidth usages on SLs actually vary much faster, and thus using VNT reconfiguration to balance them would induce much more frequent reconfigurations and complicate NC&M to an unbearable level. Secondly, we have other options to balance the bandwidth usages in much simpler ways, *e.g.*, applying traffic engineering techniques in each VNT [41].

Line 1 of *Algorithm* 1 is for preprocessing, and it selects the VMs that are running on heavy-loaded racks and thus should be migrated. Here, the VM selection can be achieved with a trivial greedy algorithm based on an empirically-determined selection ratio. For instance, the selection algorithm designed in [25] first sorts the racks in descending order of their IT resource usages, then sequentially selects the most "critical" VMs to reconfigure such that migrating them away from their current racks can push the racks' IT resource usages close to the average value, and stores all the selected VMs in set V_R^s when the selection ratio is reached. Actually, the VM selection algorithm should be customized according to the HOE-DCN operator's expectation on the VNT reconfiguration, *e.g.*, the operator can use different selection scenarios and/or selection ratios to balance the tradeoff between the complexity and effectiveness of VNT reconfiguration. Therefore, we leave it open[1], and make sure that our algorithm design for *Lines* 2-4 can accomplish the optimization based on the selected VMs provided by the preprocessing.

In this work, we focus on designing approximation algorithms to accomplish the tasks described in *Lines* 2-4. Specifically, by migrating the selected VMs in V_R^s and remapping the related VLs, our VNT reconfiguration has the primary objective as to balance the IT resource usages in the HOE-DCN. We also consider the OXC reconfiguration's impact on optical-preferred VLs, and thus set the secondary objective as to maximize the number of optical-preferred VLs that are embedded on optical connections after the VNT reconfiguration.

In VNT reconfiguration, the VMs and VLs are the basic network elements that need to be remapped in the HOE-DCN. For load balancing, a VM can be migrated to any rack whose IT and I/O resources are enough for it, while the consequent VL remapping needs to consider the one-to-one connectivity of the OXC, if optical connections are involved. More specifically, we can reconfigure two types of VLs, *i.e.*,

[1]The simulations in Section VI still use the VM selection algorithm in [25].

Algorithm 1: Overall Procedure of VNT Reconfiguration in an HOE-DCN

1 perform preprocessing to select VMs to migrate and store the selected VMs in set V_R^s;
2 determine the remapping schemes of VMs in V_R^s;
3 calculate the reconfiguration schemes of related VLs and the OXC;
4 remap VMs in V_R^s accordingly and reconfigure all the affected VLs;

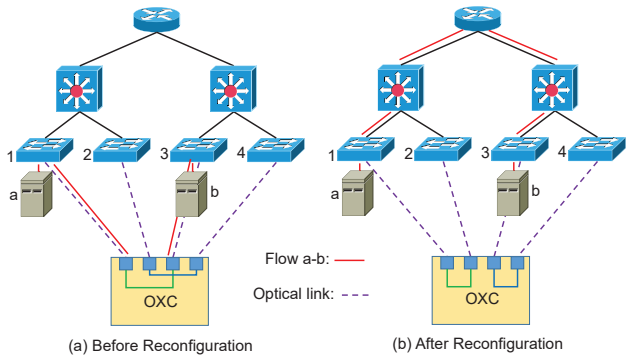


Fig. 2. Example on remapping "VLs without VM migration" to adapt to an OXC reconfiguration.

VLs with VM migration, and VLs without VM migration [25]. Here, "VLs with VM migration" refer to the VLs that have to be reconfigured to adapt to the migration of their end VMs, while "VLs without VM migration" means that the VLs need to be reconfigured purely because of an upcoming OXC reconfiguration. For example, Fig. 2 provides an illustrative example on the remapping of VLs without VM reconfiguration. Before the VNT reconfiguration (as in Fig. 2(a)), *ToR Switches* 1 and 3 can talk with each other using an optical connection through the OXC, which means that the VL (a, b) between *VMs* a and b is mapped onto the SL that represents the optical connection. Then, we reconfigure the OXC to make *ToR Switches* 1 and 2 mutually connected through it (as in Fig. 2(b)). This removes the optical connection between *ToR Switches* 1 and 3, and affects the operation of VL (a, b). Therefore, we have to remap the VL in the EPS-based inter-rack network.

IV. MILP Model for VNT Reconfiguration

After the preprocessing, we need to determine the reconfiguration schemes of the selected VMs, related VLs and OXC, which can be described with the following MILP model.

Notations:

- V_s: set of racks in the SNT.
- R: set of VNTs in the SNT.
- V_r: set of VMs in a VNT $r \in R$.
- R_s: set of rack pairs in the SNT.
- C_{v_s}: total IT capacity of servers in rack $v_s \in V_s$.
- B_{v_s}: total I/O capacity of servers in rack $v_s \in V_s$.
- $B_{(u_s,v_s)}$: total bandwidth capacity of the optical connection between a rack pair $(u_s, v_s) \in R_s$.
- c_{v_s}: IT utilization in rack $v_s \in V_s$ before reconfiguration.
- b_{v_s}: I/O utilization in rack $v_s \in V_s$ before reconfiguration.
- V_R^s: set of VMs that are chosen for reconfiguration.
- m_{v_r}: boolean that equals 1 if VM v_r is selected for reconfiguration, and 0 otherwise.
- E_r^o: set of optical-preferred VLs.
- c_{v_r}: IT usage of VM $v_r \in V_R^s$.
- b_{v_r}: I/O usage of VM $v_r \in V_R^s$.
- $\delta_{v_s}^{v_r}$: boolean that equals 1 if VM v_r is embedded on rack v_s before reconfiguration, and 0 otherwise.
- $f_{(u_s,v_s)}$: boolean that equals 1 if the OXC connects racks u_s and v_s before reconfiguration, and 0 otherwise.

Variables:

- $\widetilde{c}_{v_s}$: IT utilization in rack $v_s \in V_s$ after reconfiguration.
- $\widetilde{\delta}_{v_s}^{v_r}$: boolean that equals 1 if VM v_r is embedded on rack v_s after reconfiguration, and 0 otherwise.
- $\widetilde{f}_{(u_s,v_s)}$: boolean that equals 1 if the OXC connects racks u_s and v_s after reconfiguration, and 0 otherwise.
- $\widetilde{l}_{(u_s,v_s)}^{(u_r,v_r)}$: boolean that equals 1 if optical-preferred VL $(u_r, v_r) \in E_r^o$ is embedded on the optical link between racks u_s and v_s after reconfiguration, and 0 otherwise.
- $\widetilde{q}_{(u_r,v_r)}$: boolean that equals 1 if optical-preferred VL $(u_r, v_r) \in E_r^o$ is embedded on an optical connection after reconfiguration, and 0 otherwise.
- $c_{\max}$: the maximum ratio of IT utilization on a rack after reconfiguration.

Objective:

The primary objective is to balance the IT resource usages in the HOE-DCN, which can be realized by minimizing the maximum ratio of IT utilization on a rack after reconfiguration ($c_{\max}$). For the second objective, we can obtain the number of optical-preferred VLs that are embedded on optical connections after reconfiguration as

$$\tilde{n} = \frac{1}{2} \sum_{(u_r,v_r) \in E_r^o} \widetilde{q}_{(u_r,v_r)}. \tag{1}$$

Hence, the overall optimization objective is defined as

$$\text{Minimize} \quad (\alpha \cdot c_{\max} - \beta \cdot \tilde{n}), \tag{2}$$

where α and β are positive coefficients to weight the importance of the two objectives, and we have $\alpha \gg \beta$.

Constraints:

- *VM Mapping Constraints:*

$$\sum_{v_s \in V_s} \widetilde{\delta}_{v_s}^{v_r} = 1, \quad \forall v_r \in V_R^s. \tag{3}$$

Eq. (3) ensures that each VM still gets mapped onto one and only one rack after reconfiguration.

$$\begin{aligned} &\widetilde{\delta}_{v_s}^{v_r} \cdot (1 - m_{v_r}) - \delta_{v_s}^{v_r} \cdot (1 - m_{v_r}) = 0, \\ &\forall v_r \in V_r, \ \forall v_s \in V_s, \ r \in R. \end{aligned} \tag{4}$$

Eq. (4) ensures that after reconfiguration, the VM that does not need to be migrated is still mapped on its original rack.

- *Optical-preferred VL Mapping Constraints:*

$$\begin{aligned} &\widetilde{l}_{(u_s,v_s)}^{(u_r,v_r)} \leq \frac{\widetilde{f}_{(u_s,v_s)} + \widetilde{\delta}_{u_s}^{u_r} + \widetilde{\delta}_{u_s}^{u_r}}{3}, \\ &\forall (u_s, v_s) \in R_s, \ \forall (u_r, v_r) \in E_r^o. \end{aligned} \tag{5}$$

Eq. (5) ensures that if an optical-preferred VL needs to be embedded between a rack pair that is connected with an optical connection, the VL can be mapped on the optical connection.

$$\widetilde{q}_{(u_r,v_r)} \leq \sum_{(u_s,v_s)\in R_s} \widetilde{l}^{(u_r,v_r)}_{(u_s,v_s)},\ \forall (u_r,v_r) \in E^o_r. \tag{6}$$

Eq. (6) ensures that if an optical-preferred VL is mapped on an optical connection, it is denoted correctly.

- *OXC Reconfiguration Constraints:*

$$\sum_{\{v_s:(u_s,v_s)\in R_s\}} \widetilde{f}_{(u_s,v_s)} = 1,\ \forall u_s \in V_s, \tag{7}$$

$$\sum_{\{u_s:(u_s,v_s)\in R_s\}} \widetilde{f}_{(u_s,v_s)} = 1,\ \forall v_s \in V_s, \tag{8}$$

$$\widetilde{f}_{(u_s,v_s)} = \widetilde{f}_{(v_s,u_s)},\ \forall (u_s,v_s) \in R_s. \tag{9}$$

Eqs. (7)-(9) ensure that each rack can only talk with only one other rack through the OXC.

$$|V_s| - \sum_{(u_s,v_s)\in R_s} \widetilde{f}_{(u_s,v_s)} \cdot f_{(u_s,v_s)} \leq \eta. \tag{10}$$

Eq. (10) ensures that the total number of reconfigured ports in the OXC cannot exceed the preset non-negative threshold η.

- *Resource Constraints:*

$$c_{v_s} + \sum_{v_r\in V^s_R} c_{v_r} \cdot (\widetilde{\delta}^{v_r}_{v_s} - \delta^{v_r}_{v_s}) \leq C_{v_s},\ \forall v_s \in V_s. \tag{11}$$

Eq. (11) ensures that the IT resource utilization on each rack does not exceed its IT capacity after reconfiguration.

$$b_{v_s} + \sum_{v_r\in V^s_R} b_{v_r} \cdot (\widetilde{\delta}^{v_r}_{v_s} - \delta^{v_r}_{v_s}) \leq B_{v_s},\ \forall v_s \in V_s. \tag{12}$$

Eq. (12) ensures that the I/O resource utilization on each rack does not exceed its I/O capacity after reconfiguration. Note that, the I/O usage of a VM is the total bandwidth usage of all the VLs that end at it.

$$\widetilde{c}_{v_s} = c_{v_s} + \sum_{v_r\in V^s_R} c_{v_r} \cdot (\widetilde{\delta}^{v_r}_{v_s} - \delta^{v_r}_{v_s}),\ \forall v_s \in V_s. \tag{13}$$

Eq. (13) calculates the IT resource utilization on each rack after reconfiguration.

$$\sum_{(u_s,v_s)\in E^o_r} b_{(u_r,v_r)} \cdot \widetilde{l}^{(u_r,v_r)}_{(u_s,v_s)} \leq B_{(u_s,v_s)},\ \forall (u_s,v_s) \in R_s. \tag{14}$$

Eq. (14) ensures that bandwidth usage of each optical connection through the OXC does not exceed its bandwidth capacity after reconfiguration.

$$c_{\max} \geq \frac{\widetilde{c}_{v_s}}{C_{v_s}},\ \forall v_s \in V_s, \tag{15}$$

Eq. (15) calculates the value of $c_{\max}$.

Complexity Analysis:

Lemma 1. *The VNT reconfiguration described by the aforementioned MILP model is an $\mathcal{NP}$-hard problem.*

Proof: We prove the $\mathcal{NP}$-hardness of the problem by restriction, *i.e.*, restricting away some of its aspects until a known $\mathcal{NP}$-hard problem shows up [42].

First of all, we apply the restriction that the preset threshold on the total number of reconfigured ports in the OXC should be $\eta = 0$ in Eq. (10). This means that we do not allow any OXC reconfiguration. Then, we divide the problem solving into two steps, 1) calculating the VM migration schemes for re-balancing the loads on the racks, and 2) determining the reconfiguration schemes of related VLs. By treating each rack as a bin and each VM as an item, we can easily verify that the optimization in the first step is the general case of the load-balanced bin packing problem, which is known to be $\mathcal{NP}$-hard [43]. For the second step, if we consider each optical connection through the OXC as a knapsack and the optical-preferred VLs that can be embedded on the optical connection as items, the optimization is transformed into the general case of the knapsack problem, which is also $\mathcal{NP}$-hard [42]. Because a special/restricted case of the optimization in the MILP model is a combination of the general cases of two known $\mathcal{NP}$-hard problems, we prove its $\mathcal{NP}$-hardness. ■

V. Approximation Algorithms

As the problem of VNT reconfiguration in HOE-DCNs is $\mathcal{NP}$-hard, we try to solve it time-efficiently with polynomial-time approximation algorithms. We divide the problem solving into two steps [25], 1) calculating the migration schemes of selected VMs, and 2) obtaining the reconfiguration schemes of related VLs and the OXC[2].

A. Determining VM Migration Schemes

For the first step, the optimization to determine the migration schemes of selected VMs can be formulated as follows.

$$\begin{aligned} \text{Minimize} \quad & c_{\max}, \\ \text{s.t.} \quad & \text{Eqs. (3), (11)-(13), and (15).} \end{aligned} \tag{16}$$

In the proof of *Lemma* 1, we have already verified the $\mathcal{NP}$-hardness of this optimization. Therefore, we design an approximation algorithm for it as explained in *Algorithm* 2.

The approximation algorithm leverages linear relaxation with randomized rounding. We first obtain a linear programming (LP) model by relaxing all the boolean variables to real ones in $[0, 1]$ (*Line* 1). Then, *Line* 2 solves the LP and gets the objective Z_{LP}. With the LP's solution, we calculate the ratio of IT resource usage of each rack $v_s \in V_s$ as $\frac{\widetilde{c}_{v_s}}{C_{v_s}}$, and store the racks in set V in ascending order of their ratios (*Line* 3). The while-loop covering *Lines* 5-14 performs randomized rounding on the LP's solution (for M_1 iterations at most). Here, we first perform randomized rounding on the real variables in $\{\widetilde{\delta}^{v_r}_{v_s}\}$ with *Algorithm* 3, and obtain an integer solution $\mathbf{F}$ (*Line* 6). Then, in *Line* 7, we get the objective Z^* with $\mathbf{F}$. If $\mathbf{F}$ is a feasible solution to the original ILP, we check whether Z^* satisfied the condition of $\frac{Z^*}{Z_{\text{LP}}} \leq 1+\gamma_1$ (*Line* 9), where $(1+\gamma_1)$ is the pre-defined approximation ratio with $\gamma_1 > 0$. If yes, we stop the iterations and output $\mathbf{F}$ as the solution of the ILP.

The detailed procedure of the randomized rounding in *Line* 6 of *Algorithm* 2 is explained in *Algorithm* 3. *Lines* 1-4 are

[2]As we solve the problem in two sequential steps, the final result provided by the approximation algorithms designed in this Section might not ensure a strict gap to the exact solution from the MILP model. The approximation algorithm for the overall optimization will be studied in our future work.

Algorithm 2: Approximation Algorithm for Determining VM Migration Schemes

1 relax ILP in Eq. 16 to obtain an LP;
2 solve the LP to obtain $\{\widetilde{\delta}_{v_s}^{v_r}\}$ and its objective Z_{LP};
3 store racks in V_s in set V in ascending order of their ratios of IT resource usages;
4 $m = 1$;
5 **while** $m \leq M_1$ **do**
6 use *Algorithm* 3 to perform randomized rounding on $\{\widetilde{\delta}_{v_s}^{v_r}\}$ (based on the sorted order in V) and obtain an integer solution **F**;
7 calculate objective Z^* with **F**;
8 **if** ***F*** *is a feasible solution of original ILP* **then**
9 **if** $\frac{Z^*}{Z_{LP}} \leq 1 + \gamma_1$ **then**
10 **break**;
11 **end**
12 **end**
13 $m = m + 1$;
14 **end**
15 **return F** and Z^*;

Algorithm 3: Randomized Rounding

Input: $\{\widetilde{\delta}_{v_s}^{v_r}\}$, V
Output: **F**.
1 $\mathbf{F} = \emptyset$;
2 set variable $x_{v_r} = 1$ for each selected VMs;
3 generate a random number p within $(0, 1)$;
4 calculate the IT usage $\widetilde{c}_{u_s}^t$ on each $u_s \in V_s$ by removing all the VMs selected to migrate;
5 **for** *each* $v_s \in V$ *in sorted order* **do**
6 **for** *each VM* v_r *with* ($x_{v_r} = 1$ *and* $\widetilde{\delta}_{v_s}^{v_r} \geq p$) **do**
7 insert $\widetilde{\delta}_{v_s}^{v_r} = 1$ in **F**;
8 $x_{v_r} = 0$;
9 **end**
10 **end**
11 **for** *each VM* v_r *with* ($x_{v_r} = 1$) **do**
12 $v_s = \underset{\{u_s \in V_s\}}{\text{argmin}} \left(\frac{\widetilde{c}_{u_s}^t}{C_{u_s}}\right)$;
13 insert $\widetilde{\delta}_{v_s}^{v_r} = 1$ in **F** and update $\widetilde{c}_{u_s}^t$;
14 $x_{v_r} = 0$;
15 **end**

for the initialization. In the for-loop that covers *Lines* 5-10, we check all the racks in set V in ascending order of their ratios of IT resource usages. For each rack v_s, we find all the VMs that satisfy $x_{v_r} = 1$ and $\widetilde{\delta}_{v_s}^{v_r} \geq p$ (*Line* 6), round the corresponding variables ($\{\widetilde{\delta}_{v_s}^{v_r}\}$) to 1 and insert them in **F** (*Line* 7), and label these VMs with $x_{v_r} = 0$ to denote that their integer solutions have been obtained (*Line* 8). Next, the integer solutions for the remaining VMs (*i.e.*, those still with $x_{v_r} = 1$) are obtained with the for-loop covering *Lines* 11-15.

The time complexity of *Algorithm* 3 is $O(|V| \cdot |V_R^s|)$. For *Algorithm* 2, the LP can be solved in polynomial-time, *e.g.*, the time complexity is $O(X^{3.5} \cdot Y)$ when we use the interior point method [44], where X is the number of variables in the LP and Y is the total number of bits of the input. Hence, the complexity of *Algorithm* 2 is $O(M_1 \cdot |V| \cdot |V_R^s| + X^{3.5} \cdot Y)$, which indicates that it is a polynomial-time algorithm.

Meanwhile, we can easily verify that the approximation ratio of *Algorithm* 2 is upper-bounded by $(1 + \gamma_1)$ as follows. Since the ILP in Eq. 16 is for minimization, the Z_{LP} and Z^* obtained with *Algorithm* 2 are the lower- and upper-bounds of its optimal solution (denoted as Z_{ILP}), respectively. Hence, we can calculate the approximation ratio of *Algorithm* 2 as

$$\epsilon = \frac{Z^*}{Z_{\text{ILP}}} \leq \frac{Z^*}{Z_{\text{LP}}} \leq 1 + \gamma_1. \tag{17}$$

Finally, we would like to explain that according to the principle of linear relaxation with randomized rounding and the well-known Chernoff-Bound [45], the probability of *Algorithm* 2 finding a qualified feasible solution can approach to 1, as long as the values of M_1 and γ_1 are properly selected. We will show the convergence performance of *Algorithm* 2 in Section VI.

B. Obtaining Reconfiguration Schemes of OXC and VLs

In the second step, we need to obtain the reconfiguration schemes of the OXC and related VLs based on the VM migration schemes determined above, to maximize the number of optical-preferred VLs that are embedded on optical connections after reconfiguration. This problem can be solved with *Algorithm* 4 [25]. Since we already know the VM migration schemes at this moment, all the rack pairs that will be connected with inter-rack VLs after reconfiguration should also be known. We store these rack pairs in set R_s.

Algorithm 4: Obtaining Reconfiguration Schemes of OXC and Related VLs

1 **for** *each rack pair* $(u_s, v_s) \in R_s$ **do**
2 use dynamic programming in [25] to get the largest number of optical-preferred VLs (n_{u_s,v_s}) that optical connection for $u_s \leftrightarrow v_s$ can carry;
3 obtain the corresponding VL mapping schemes;
4 **end**
5 use *Algorithm* 6 to get the reconfiguration schemes of OXC and related VLs;

Lines 1-4 check the rack pairs in R_s. Specifically, for each rack pair (u_s, v_s), we assume that the two racks have an optical connection through the OXC, calculate the largest number of optical-preferred VLs that the optical connection can accommodate (*i.e.*, n_{u_s,v_s}), and determine the remapping schemes for the VLs accordingly. Here, the optical-preferred VLs, which should be mapped on the optical connection between a rack pair $(u_s, v_s) \in R_s$ to ensure that the optical connection carries the largest number of optical-preferred VLs, can be find with the linear-time dynamic programming developed in [25]. Finally, based on the results from the dynamic programming, we leverage *Algorithm* 6 to obtain the reconfiguration schemes of the OXC and related VLs (*Line* 5).

The optimization that should be tackled with *Algorithm* 6

can be summarized as the following ILP model.

$$\begin{aligned} \text{Maximize} \quad & \frac{1}{2} \sum_{(u_s,v_s)\in R_s} n_{u_s,v_s} \cdot \widetilde{f}_{(u_s,v_s)}, \\ \text{s.t.} \quad & \text{Eqs. (7)-(10).} \end{aligned} \tag{18}$$

We utilize Lagrangian relaxation to propose a polynomial-time approximation algorithm for this ILP model as follows.

1) Constructing Lagrangian Dual Problem: We first dualize the constraint in Eq. (10) and construct the following dual problem, whose solution gives an upper-bound on the optimal solution of the ILP in Eq. (18).

$$\begin{aligned} \text{Minimize } Z_{\text{dual}}(\lambda) = \max_{\{\widetilde{f}_{(u_s,v_s)}\}} & \left[\left(\frac{1}{2} \sum_{(u_s,v_s)\in R_s} n_{u_s,v_s} \cdot \widetilde{f}_{(u_s,v_s)} \right) \right. \\ & \left. + \lambda \cdot \left(\eta - |V_s| + \sum_{(u_s,v_s)\in R_s} \widetilde{f}_{(u_s,v_s)} \cdot f_{(u_s,v_s)} \right) \right], \\ \text{s.t.} \quad & \text{Eqs. (7)-(9),} \end{aligned} \tag{19}$$

where $\lambda \geq 0$ is the Lagrangian multiplier. As we need to maximize $Z_{\text{dual}}(\lambda)$ for a specific λ, the dual problem becomes

$$\begin{aligned} \text{Minimize } Z_{\text{dual}}(\lambda) = \max_{\{\widetilde{f}_{(u_s,v_s)}\}} & \left[\left(\frac{1}{2} \sum_{(u_s,v_s)\in R_s} \bar{n}_{u_s,v_s} \cdot \widetilde{f}_{(u_s,v_s)} \right) \right. \\ & \left. + \lambda \cdot (\eta - |V_s|) \right], \\ \text{s.t.} \quad & \text{Eqs. (7)-(9),} \end{aligned} \tag{20}$$

where $\bar{n}_{u_s,v_s}$ is the Lagrangian-modified number of optical-preferred VLs that are embedded on the optical connection between the rack pair (u_s, v_s).

$$\bar{n}_{u_s,v_s} = n_{u_s,v_s} + 2\lambda \cdot f_{(u_s,v_s)}. \tag{21}$$

We further modify the optimization in Eq. (20) by deleting the last constraint (Eq. (9)), and because the term $\lambda \cdot (\eta - |V_s|)$ in Eq. (20) is independent of $\{\widetilde{f}_{(u_s,v_s)}\}$, they can be removed too. Then, the optimization is modified to

$$\begin{aligned} \text{Maximize} \quad & \frac{1}{2} \sum_{(u_s,v_s)\in R_s} \bar{n}_{u_s,v_s} \cdot \widetilde{f}_{(u_s,v_s)}, \\ \text{s.t} \quad & \sum_{\{v_s:(u_s,v_s)\in R_s\}} \widetilde{f}_{(u_s,v_s)} \leq 1, \ \forall u_s \in V_s, \\ & \sum_{\{u_s:(u_s,v_s)\in R_s\}} \widetilde{f}_{(u_s,v_s)} \leq 1, \ \forall v_s \in V_s. \end{aligned} \tag{22}$$

Lemma 2. *The problem in Eq.* (22) *is equivalent to that of finding the maximal weight matching in a bipartite graph.*

Proof: We first construct a bipartite graph that consists of two sets A and B, each of which includes $|V_s|$ elements. Then, we define a_i and b_i as the i-th elements in A and B, respectively. Next, we set the weight of the connection between a_i and b_j as $w_{i,j}$. The boolean $x_{(i,j)}$ is defined to be 1 if a_i and b_j are connected in the bipartite graph, and 0 otherwise. Therefore, the ILP model to obtain the maximal weight matching in the bipartite graph can be formulated as

$$\begin{aligned} \text{Maximize} \quad & \sum_{i,j\in[1,|V_s|]} w_{i,j} \cdot x_{(i,j)}, \\ \text{s.t} \quad & \sum_{i\in[1,|V_s|]} x_{(i,j)} \leq 1, \ \forall j \in [1,|V_s|], \\ & \sum_{j\in[1,|V_s|]} x_{(i,j)} \leq 1, \ \forall i \in [1,|V_s|], \end{aligned} \tag{23}$$

which shares the same formulation of Eq. (22) if we replace $w_{i,j}$ and $x_{(i,j)}$ with $\frac{1}{2} \cdot \bar{n}_{u_s,v_s}$ and $\widetilde{f}_{(u_s,v_s)}$, respectively. ■

Note that, the maximal weight matching in a bipartite graph can be found with the Kuhn-Munkres algorithm [46], whose complexity is $O(|V_s|^3)$ to solve the optimization in Eq. (22).

We take the maximal value of the problem depicted in Eq. (22), add $\lambda \cdot \eta - \lambda \cdot |V_s|$ to it to get $Z^*_{\text{dual}}(\lambda)$, and finally get the following dual problem

$$\begin{aligned} \text{Minimize } Z^*_{\text{dual}}(\lambda) = \max_{\{\widetilde{f}_{(u_s,v_s)}\}} & \left[\left(\frac{1}{2} \sum_{(u_s,v_s)\in R_s} n_{u_s,v_s} \cdot \widetilde{f}_{(u_s,v_s)} \right) \right. \\ & \left. + \lambda \cdot \left(\eta - |V_s| + \sum_{(u_s,v_s)\in R_s} \widetilde{f}_{(u_s,v_s)} \cdot f_{(u_s,v_s)} \right) \right], \\ \text{s.t} \quad & \sum_{\{v_s:(u_s,v_s)\in R_s\}} \widetilde{f}_{(u_s,v_s)} \leq 1, \ \forall u_s \in V_s, \\ & \sum_{\{u_s:(u_s,v_s)\in R_s\}} \widetilde{f}_{(u_s,v_s)} \leq 1, \ \forall v_s \in V_s. \end{aligned} \tag{24}$$

Although the one in Eq. (24) is not the strict Lagrangian dual problem of the original problem in Eq. (18), we always have $Z_{\text{dual}}(\lambda) \leq Z^*_{\text{dual}}(\lambda)$ due to the expanded solution space. Hence, $Z^*_{\text{dual}}(\lambda)$ still provides an upper-bound on the optimal solution of the original problem.

2) Solving Lagrangian Dual Problem: The optimization in Eq. (24) is a piecewise LP, which can be solved by leveraging the sub-gradient method in [47] to update λ iteratively until $Z^*_{\text{dual}}(\Lambda)$ converges to the minimum. We update λ as follows.

$$\lambda_{k+1} = \lambda_k - \mu_k \cdot f(\lambda_k), \tag{25}$$

where μ_k and $f(\lambda_k)$ are the step-size and sub-gradient vector of $Z^*_{\text{dual}}(\lambda)$ regarding λ, respectively, for the k-th iteration. The sub-gradient vector can be obtained as

$$f(\lambda) = \frac{\partial Z^*_{\text{dual}}}{\partial \lambda} = \eta - |V_s| + \sum_{(u_s,v_s)\in R_s} \widetilde{f}_{(u_s,v_s)} \cdot f_{(u_s,v_s)}. \tag{26}$$

As the value of step-size μ_k affects the convergence performance, we determine it as follows, according to [48].

$$\mu_k = \frac{\nu \cdot (Z_{\text{dual}}(\lambda_k) - Z^*)}{||f(\lambda_k)||^2}, \tag{27}$$

where $Z_{\text{dual}}(\lambda_k)$ is obtained by solving the problem in Eq. (22) with a specific Lagrangian multiplier λ_k, Z^* is the maximal feasible solution until the k-th iteration, and ν is a variable whose initial value is 2. Note that, if the value of $Z_{\text{dual}}(\lambda_k)$ does not reduce after a fixed number of iterations, we divide ν by 2. To ensure that $Z_{\text{dual}}(\lambda)$ is an upper-bound on the optimal solution, we need to have $\lambda \geq 0$, which is achieved with

$$\lambda_{k+1} = \max\{0, \ [\lambda_k - \mu_k \cdot f(\lambda_k)]\}. \tag{28}$$

3) Obtaining Feasible Solution: Then, we design a heuristic to find a feasible solution of the original problem in the ILP in Eq. (18), and the solution provides a lower-bound on the exact solution in each iteration. The heuristic is shown in *Algorithm* 5, which obtains a feasible solution $\{\widetilde{f}^k_{(u_s,v_s)}\}$ and updates Z^* in the k-th iteration. In *Line* 1, we initialize the counter x as 0, and the temporary variables $\{\widetilde{f}^t_{(u_s,v_s)}\}$ as the solution obtained in the $(k-1)$-th iteration, *i.e.*, $\{\widetilde{f}^{k-1}_{(u_s,v_s)}\}$. The while-loop covering *Lines* 2-26 tries to obtain a solution that is better than $\{\widetilde{f}^{k-1}_{(u_s,v_s)}\}$. The while-loop runs for Q times at most, and we will study the effect of the value of Q with simulations in Section VI. Specifically, in *Lines* 6-15, we check the connection scheme of four OXC ports each at a time, and determine that whether reconfiguring their connection scheme leads to a better solution or not. Then, *Lines* 16-25 update the OXC reconfiguration scheme to include the connection scheme of four OXC ports, which results in the largest increase over the previous feasible solution $\{\widetilde{f}^{k-1}_{(u_s,v_s)}\}$. At last, in *Line* 27, we get a better lower-bound Z^* with the new feasible solution $\{\widetilde{f}^{k}_{(u_s,v_s)}\}$. The time complexity of *Algorithm* 5 is $O(Q\cdot|V_s|^2)$.

4) Overall Procedure: Finally, the proposed approximation algorithm that leverages Lagrangian relaxation to solve the ILP in Eq. (18) is explained in *Algorithm* 6. Specifically, by solving the Lagrangian dual problem (*i.e.*, the upper-bound) and obtaining feasible solutions with the heuristic in *Algorithm* 5 (*i.e.*, the lower-bound), we optimize the solution iteratively. *Line* 1 is for the initialization, where ub and lb are for the upper- and lower-bounds obtained in each iteration, respectively, and n is the counter to monitor the convergence condition of $Z^*_{\text{dual}}(\lambda)$. Then, the while-loop that covers *Lines* 2-20 optimizes the solution until its approximation ratio is greater than a preset threshold $(1-\gamma_2)$ (*Lines* 14-16). In *Lines* 5-11, we update the upper-bound ub with $Z^*_{\text{dual}}(\lambda)$, and if ub stays as unchanged for T_h iterations, we divide v by 2. *Lines* 17-18 calculate μ_k and λ_{k+1} to prepare for the next iteration. The while-loop will run for M_2 iterations at most, and thus the time complexity of *Algorithm* 6 is $O(M_2\cdot|V_s|^2\cdot(Q+|V_s|))$.

As the original problem in the ILP in Eq. (18) is for maximization, we can prove that the approximation ratio of *Algorithm* 6 is lower-bounded by $\gamma_2 \in (0,1)$ as follows. We still define the optimal solution as Z_{ILP}, and then the approximation ratio for the maximization problem is

$$\epsilon = \frac{Z^*}{Z_{\text{ILP}}} \geq \frac{Z^*}{Z^*_{\text{dual}}(\lambda)} \geq 1-\gamma_2. \tag{29}$$

This is because the $Z^*_{\text{dual}}(\lambda)$, which is obtained by solving the dual problem, provides the upper-bound on Z_{ILP}. We also want to point out that according to the principle of Lagrangian relaxation [47], *Algorithm* 6 can converge and find a qualified feasible solution as long as the values of M_2 and γ_2 are properly selected. The actual convergence performance of the algorithm will be discussed in the next section.

VI. Performance Evaluations

In this section, we perform numerical simulations to evaluate the performance of our proposed algorithms.

Algorithm 5: Obtaining Feasible Solution

Input: η, Q, $\{\widetilde{f}^{k-1}_{(u_s,v_s)}\}$, $\{n_{u_s,v_s}\}$.
Output: Z^*.

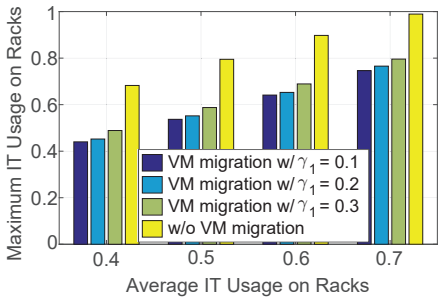
1 $x=0$, $\{\widetilde{f}^t_{(u_s,v_s)}\}=\{\widetilde{f}^{k-1}_{(u_s,v_s)}\}$;
2 **while** $x \leq Q$ **do**
3 $x=x+1$, $y=0$, $\kappa_1=\kappa_2=0$;
4 assign indices of existing optical connections in $\{\widetilde{f}^t_{(u_s,v_s)}\}$ as $\{l_i,\ i\in\left[1,\frac{|V_s|}{2}\right]\}$;
5 store the optical connections in set L;
6 **for** $i \in \left[1,\frac{|V_s|}{2}-1\right]$ **do**
7 **for** $j \in \left[i+1,\frac{|V_s|}{2}\right]$ **do**
8 $s_{ori}=n_i+n_j$;
9 get the four OXC ports of l_i and l_j;
10 find the OXC ports' connection scheme that the largest number of optical-preferred VLs (s_{max}) can be embedded on the resulting optical connections;
11 **if** $s_{max}-s_{ori}>y$ **then**
12 $y=s_{max}-s_{ori}$, $\kappa_1=i$, $\kappa_2=j$;
13 **end**
14 **end**
15 **end**
16 **if** $y>0$ **then**
17 reconfigure the OXC ports of l_{κ_1} and l_{κ_2} and update $\{\widetilde{f}^t_{(u_s,v_s)}\}$ accordingly;
18 **if** $|V_s|-\sum_{(u_s,v_s)\in R_s}\widetilde{f}^t_{(u_s,v_s)}\cdot f_{(u_s,v_s)} \leq \eta$ **then**
19 $\{\widetilde{f}^k_{(u_s,v_s)}\}=\{\widetilde{f}^t_{(u_s,v_s)}\}$;
20 **else**
21 **break**;
22 **end**
23 **else**
24 **break**;
25 **end**
26 **end**
27 calculate Z^* according to $\{\widetilde{f}^k_{(u_s,v_s)}\}$;

A. Simulation Setup

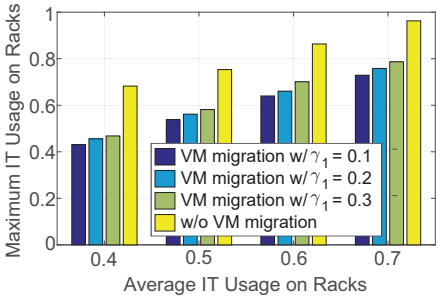
The simulations architect the EPS-based inter-rack network in the HOE-DCN with the well-known k-ray fat-tree topology [49], where there are $\frac{k^2}{2}$ racks/ToR switches evenly distributed in k points-of-delivery (PoDs). Each ToR switch is equipped with $\frac{k}{2}$ Ethernet ports, which are connected to its aggregation switches, and one optical port that is connected to the OXC. To evaluate our approximation algorithms in depth, we surveyed commercially-available large-scale OXCs, and found that those with the configuration of 384×384 ports are commonly used (*e.g.*, the Polatis Series 7000 [50]). Hence, the largest HOE-DCN considered in the simulations uses the 28-ray fat-tree topology that includes 392 racks. The simulation parameters are either adopted from real-world DCNs or based on the observations in our experimental demonstrations in [9, 10], and thus the choices are practical and can represent the cases

Algorithm 6: Approximation Algorithm for Obtaining Reconfiguration Schemes of OXC and Related VLs

$k = 1$, $\lambda_k = 0$, $\nu = 2$, $ub = +\infty$, $lb = 0$, $n = 0$;
while $k \leq M_2$ **do**
  calculate $\{\bar{n}_{u_s,v_s}\}$ with Eq. (21) and λ_k;
  solve optimization in Eq. (22) with Kuhn-Munkres algorithm for $Z^*_{\text{dual}}(\lambda_k)$ and $\{\widetilde{f}_{(u_s,v_s)}\}$;
  if $Z^*_{dual}(\lambda_k) < ub$ **then**
    $ub = Z_{\text{dual}}(\lambda_k)$, $n = 0$;
  else if $n > T_h$ **then**
    $\nu = \nu/2$, $n = 0$;
  else
    $n = n + 1$;
  end
  get a feasible solution and Z^* with *Algorithm* 5;
  $lb = Z^*$;
  if $\frac{lb}{ub} \geq 1 - \gamma_2$ **then**
    break;
  end
  calculate μ_k with Eq. (27);
  calculate λ_{k+1} with Eqs. (25)-(28);
  $k = k + 1$;
end



(a) HOE-DCN with 20-ray fat-tree



(b) HOE-DCN with 28-ray fat-tree

Fig. 3. Maximum IT usage on racks after VM migration.

in realistic HOE-DCNs. We set the bandwidth capacity of each Ethernet port on a ToR switch as 1000 units, while that of its optical port is assumed to be 10000 units. For the k-ray fat-tree topology used in the simulations, we assume that the IT resource capacity of each rack in it is $1000 \cdot \frac{k}{2}$ units.

We use the Poisson model to generate VNTs dynamically with random topologies[3]. The number of VMs in each VNT are selected within $[2, 40]$ and $[2, 60]$ for HOE-DCNs with 20-ray and 28-ray fat-trees, respectively, and the connectivity ratio of the VMs in each VNT is set as 0.5. Both the IT and I/O resource demands of a VM are selected randomly within $[250, 1000]$ units. In the VNTs, we randomly select 50% of the VLs and label them as "optical-preferred" ones. At each simulation time, we use the VNE algorithm developed in [27] to serve new VNTs, and release the resources occupied by the expired ones. Then, we pause the VNT provisioning to invoke a VNT reconfiguration, when the IT resource usages in the HOE-DCN become unbalanced, *i.e.*, the number of "hot-spot" racks whose IT resource usages are above the average value exceeds a preset threshold. In order to maintain sufficient statistical accuracy, our simulations average the results from 5 independent runs to obtain each data point.

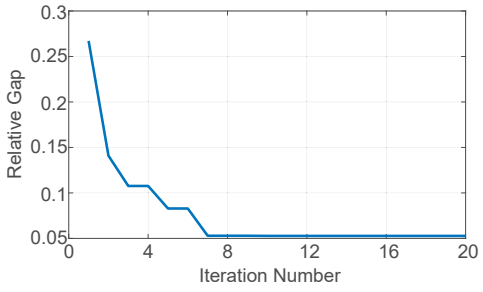
B. Feature Verification

We first conduct simulations to confirm the features of our proposed approximation algorithms. Since the problem solving of VNT reconfiguration is divided into two steps, we evaluate the algorithms designed for them one by one. In the following, we refer to the ILP models defined with Eqs. (16) and (18) as ILP-1 and ILP-2, respectively.

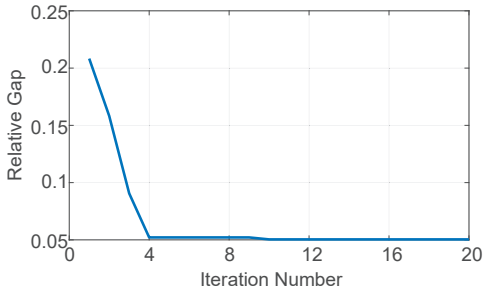
1) Determining VM Migration Schemes: To evaluate the performance of *Algorithm* 2, we invoke VNT reconfiguration in network environments where the average IT usages on the racks in an HOE-DCN are within $[0.4, 0.7]$, and set the approximation ratio as $\gamma_1 \in \{0.1, 0.2, 0.3\}$. Fig. 3 shows the results on the maximum IT usage on racks after the VM migration. We can see that for both the HOE-DCNs (with 200 and 392 racks), our VM migration algorithm effectively reduces the maximum IT usage on racks, and thus the IT resource utilizations have been re-balanced effectively. As expected, the maximum IT usage on racks can be pushed down to a lower value, if the average IT usage is smaller. It is promising to observe that our algorithm can achieve an approximation ratio of $(1+\gamma_1) = 1.1$ for the large-scale HOE-DCNs. This means that for the HOE-DCN that consists of 392 racks, the results from our approximation algorithm are at most 10% larger than the exact solutions of the minimization in Eqs. (16). We also notice that the quality of the solutions from *Algorithm* 2 improves when the value of γ_1 decreases.

Fig. 4 illustrates the worst-case convergence performance of *Algorithm* 2 (*i.e.*, the average IT usage is set as 0.7), where the relative gap is calculated as $\frac{Z^* - Z_{\text{LP}}}{Z_{\text{LP}}}$ based on Eq. (17). We observe that for both scenarios, *Algorithm* 2 reduces the relative gap to less than 0.06 within only 8 iterations. Table I lists the average running time of *Algorithm* 2 for calculating the migration scheme of each VM. Note that, due to the fact that solving ILP-1 for the large-scale HOE-DCNs is intractable, we do not list its running time here. The results in Table I indicate that the running time increases with the average IT usage. This is because we need to determine the

[3]As the design of our algorithm does not apply any assumption on the traffic model of dynamic VNTs, it should also work well when the VNTs are generated according to realistic traces measured in working DCs. Due to the page limit of the journal, we will verify this claim in our future work.



(a) HOE-DCN with 20-ray fat-tree



(b) HOE-DCN with 28-ray fat-tree

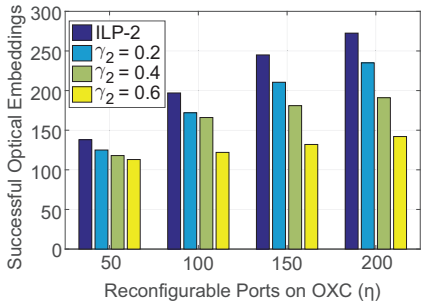
Fig. 4. Worst-case convergence performance of *Algorithm* 2.

TABLE I
AVERAGE RUNNING TIME OF *Algorithm* 2 PER VM MIGRATION (MSEC)

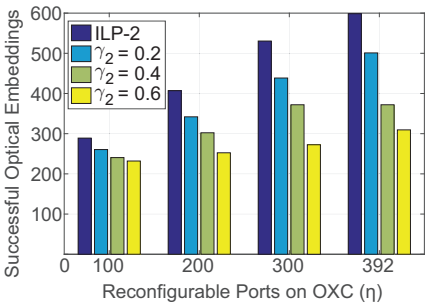
Average IT usage on racks	0.4	0.5	0.6	0.7
γ_1	20-ray Fat-tree			
0.1	4.64	5.45	6.75	7.60
0.2	4.62	5.42	6.73	7.58
0.3	4.62	5.42	6.72	7.57
γ_1	28-ray Fat-tree			
0.1	21.48	23.57	27.65	36.11
0.2	21.41	23.58	27.54	35.98
0.3	21.40	23.58	27.53	35.97

migration schemes for more VMs when the average IT usage is larger. Meanwhile, the running time stays almost unchanged when γ_1 reduces. This is because *Algorithm* 2 spends most of its running time on solving the LP, while it only runs very few more iterations to satisfy a smaller γ_1. This further confirms the superior convergence performance of our algorithm.

2) Determining Reconfiguration Schemes of OXC and Related VLs: Next, we evaluate the performance of *Algorithm* 6 on determining the reconfiguration schemes of the OXC and related VLs. Here, we set the threshold on the total number of reconfigured ports in the OXC as $\eta \in \{50, 100, 150, 200\}$ and $\eta \in \{100, 200, 300, 392\}$ for the HOE-DCNs with 20-ray and 28-ray fat-trees, respectively. The approximation ratio γ_2 is chosen from $\{0.2, 0.4, 0.6\}$. Since ILP-2 is relatively simple and can be solved within reasonable time, we also solve it to obtain the optimal solutions. Fig. 5 shows the results on the number of successful optical embeddings. Here, a successful optical embedding means that an optical-preferred VL gets mapped on an optical connection through the OXC successfully. We can see that the number of successful optical embeddings increases with the preset threshold on reconfigurable OXC ports (η). This is because a greater η provides a larger solution space. Meanwhile, the results also indicate that



(a) HOE-DCN with 20-ray fat-tree



(b) HOE-DCN with 28-ray fat-tree

Fig. 5. Embeddings of optical-preferred VLs.

TABLE II
AVERAGE RUNNING TIME OF *Algorithm* 6 AND ILP-2 (SEC)

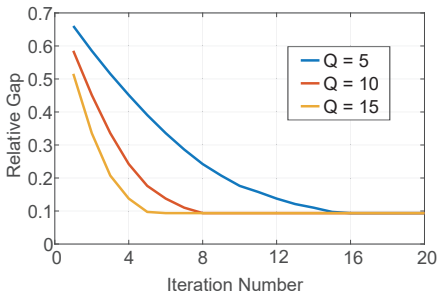
γ_2	0.2	0.4	0.6
20-ray Fat-tree			
ILP-2	12.194		
$Q = 5$	1.360	0.507	0.236
$Q = 10$	1.178	0.329	0.155
$Q = 15$	1.183	0.213	0.155
28-ray Fat-tree			
ILP-2	69.672		
$Q = 10$	7.192	3.804	1.901
$Q = 20$	6.524	2.769	0.940
$Q = 30$	5.997	2.456	0.701

we can improve the quality of solutions from *Algorithm* 6 by reducing the approximation ratio γ_2.

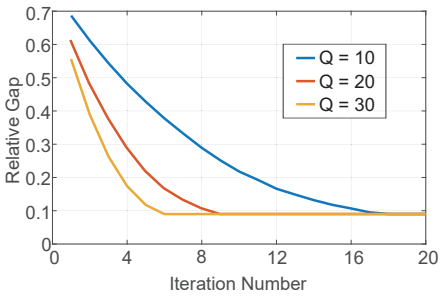
The value of Q in *Algorithm* 5 impacts the quality of feasible solutions and in turn affects the convergence performance of *Algorithm* 6. Fig. 6 illustrates the effect of Q on the convergence performance of *Algorithm* 6, which indicates that the algorithm converges faster with a larger Q. Meanwhile, for all the scenarios, the relative gap of *Algorithm* 6, which is calculated as $\frac{Z^*_{\text{dual}}(\lambda) - Z^*}{Z^*_{\text{dual}}(\lambda)}$ based on Eq. (29), can be reduced to less than 0.1 within 20 iterations. The results on the average running time of *Algorithm* 6 and ILP-2 are listed in Table II. We can see that *Algorithm* 6 is much more time-efficient than ILP-2, and the running time of *Algorithm* 6 decreases with γ_2 but increases when Q decreases. This is because a smaller Q can makes *Algorithm* 6 converge slowly as shown in Fig. 6.

C. Performance Benchmarking

Finally, we benchmark the performance of the overall procedure for VNT reconfiguration in HOE-DCNs. Specifically, by integrating *Algorithms* 2 and 6 in *Algorithm* 1, we have

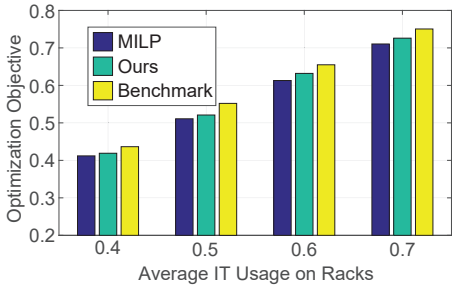


(a) HOE-DCN with 20-ray fat-tree

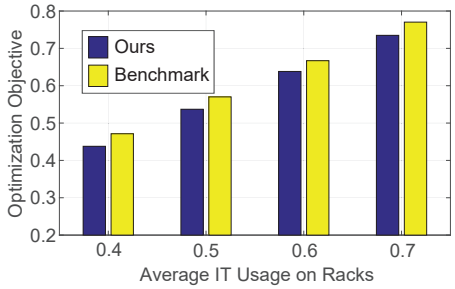


(b) HOE-DCN with 28-ray fat-tree

Fig. 6. Convergence Performance of *Algorithm* 5.



(a) HOE-DCN with 4-ray fat-tree



(b) HOE-DCN with 20-ray fat-tree

Fig. 7. Overall optimization objective.

the complete procedure and the maximum iteration numbers are set as $M_1 = M_2 = 20$. Then, we compare it with both the MILP in Section IV and the heuristic developed in [25]. Here, for the overall objective in Eq. (2), we define $\alpha = 1$ and $\beta = \frac{|V_s|}{|E_r^o| \cdot \sum_{v_s \in V_s} C_{v_s}}$, to ensure that minimizing $c_{\max}$ is the primary objective. The simulations use the 4-ray and 20-ray fat-trees for the HOE-DCNs, while we only evaluate the MILP with the 4-ray fat-tree due to its time complexity.

Fig. 7 shows the results on the overall optimization objective, where "Ours" refers to the *Algorithm* 1 that uses our proposed approximation algorithms, and "Benchmark" is for the heuristic developed in [25]. In Fig. 7(a), we can see that the MILP provides the best solution while the optimization gap of Ours is smaller than that of Benchmark. We also measure the running time of the three algorithms and list the results in Table III, which indicates that Ours uses comparable running time as Benchmark, and both of them are much more time-efficient than the MILP. Meanwhile, for the large-scale HOE-DCN with 20-ray fat-tree (*i.e.*, 200 racks), the results in Fig. 7(b) still confirm that Ours outperforms Benchmark. Moreover, the simulation results also confirm that our proposal can be implemented in a real-world HOE-DCN and adapt to the dynamic network environment in it. For instance, the results on running time in Table III suggest that Ours only uses a few milliseconds to obtain the reconfiguration schemes of tens of VNTs in an HOE-DCN with 4-ray fat-tree. In our future work, we will implement Ours in the network orchestration systems developed in [9, 10], to further verify its practicalness and evaluate its performance with experiments.

TABLE III
RUNNING TIME OF ALGORITHMS FOR HOE-DCN WITH 4-RAY FAT-TREE (SEC)

Average IT usage	0.4	0.5	0.6	0.7
MILP	0.302	1.626	10.406	267.002
Ours	5.403e-3	5.911e-3	6.842e-3	7.713e-3
Benchmark	3.306e-3	3.924e-3	4.723e-3	5.301e-3

VII. CONCLUSION

We studies how to realize effective VNT reconfiguration in an HOE-DCN such that the IT resource usages in racks can be re-balanced with VM migration. We first formulated an MILP to present the overall optimization for computing the new VNE schemes of VNTs based on preselected VMs. Then, the problem solving was into two steps, 1) calculating the VM migration schemes for the VNTs to balance the loads on racks, and 2) determining the reconfiguration schemes of related VLs and the OXC. For the first step, we proposed a polynomial-time approximation algorithm by leveraging linear relaxation. The optimization of the second step was solved by an algorithm that involves a linear-time dynamic programming and an ILP. To solve the ILP time-efficiently, we proposed another polynomial-time approximation algorithm based on Lagrangian relaxation. Our performance evaluations with extensive simulations confirmed the effectiveness of the proposed approximation algorithms, verified that they can get near-optimal solutions whose performance gaps to the optimal ones are bounded, and demonstrated that the overall procedure including them outperforms the existing approach.

ACKNOWLEDGMENTS

This work was supported in part by the NSFC projects 61871357, 61771445 and 61701472, ZTE Research Fund PA-HQ-20190925001J-1, Zhejiang Lab Research Fund 2019LE0AB01, CAS Key Project (QYZDY-SSW-JSC003), and SPR Program of CAS (XDC02070300).

REFERENCES

[1] Cisco Global Cloud Index: Forecast and Methodology, 2016-2021. [Online]. Available: https://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html

[2] 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.

[3] 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.

[4] H. Lu, M. Zhang, Y. Gui, and J. Liu, "QoE-driven multi-user video transmission over SM-NOMA integrated systems," *IEEE J. Sel. Areas Commun.*, vol. 37, pp. 2102–2116, Sept. 2019.

[5] H. Wu and H. Lu, "Delay and power tradeoff with consideration of caching capabilities in dense wireless networks," *IEEE Trans. Wireless Commun.*, vol. 18, pp. 5011–5025, Oct. 2019.

[6] 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.

[7] N. Farrington *et al.*, "Helios: a hybrid electrical/optical switch architecture for modular data centers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 339–350, Oct. 2010.

[8] 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.

[9] 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.

[10] 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.

[11] Z. Zhu *et al.*, "RF photonics signal processing in subcarrier multiplexed optical-label switching communication systems," *J. Lightw. Technol.*, vol. 21, pp. 3155–3166, Dec. 2003.

[12] 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.

[13] Z. Zhu *et al.*, "Jitter and amplitude noise accumulations in cascaded all-optical regenerators," *J. Lightw. Technol.*, vol. 26, pp. 1640–1652, Jun. 2008.

[14] 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.

[15] L. Zhang and Z. Zhu, "Spectrum-efficient anycast in elastic optical inter-datacenter networks," *Opt. Switch. Netw.*, vol. 14, pp. 250–259, Aug. 2014.

[16] C. Chen *et al.*, "Demonstrations of efficient online spectrum defragmentation in software-defined elastic optical networks," *J. Lightw. Technol.*, vol. 32, pp. 4701–4711, Dec. 2014.

[17] Z. Zhu, W. Lu, L. Liang, and B. Kong, "Predictive analytics in hybrid optical/electrical DC networks," in *Proc. of OFC 2019*, pp. 1–3, Mar. 2019.

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

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

[20] 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.

[21] 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, 2018.

[22] 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.

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

[24] 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.

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

[26] M. Chowdhury and M. Rahman, "ViNEYard : Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping," *IEEE/ACM Trans. Netw.*, vol. 20, pp. 206–219, Jan. 2012.

[27] 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.

[28] 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.

[29] A. Song *et al.*, "Distributed virtual network embedding system with historical archives and set-based particle swarm optimization," *IEEE Trans. Syst., Man, Cybern., Syst., in Press*, 2019.

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

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

[32] A. Fischer *et al.*, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, pp. 1888–1906, Fourth Quarter 2013.

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

[34] 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.

[35] S. Shaw and A. Singh, "A survey on scheduling and load balancing techniques in cloud computing environment," in *Proc. of ICCCT 2010*, pp. 87–95, Sept. 2014.

[36] N. Chien, N. Son, and H. Loc, "Load balancing algorithm based on estimating finish time of services in cloud computing," in *Proc. of ICACT 2016*, pp. 228–233, Jan. 2016.

[37] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proc. of CCGRID 2010*, pp. 826–831, May 2010.

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

[39] 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.

[40] 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.

[41] S. Li *et al.*, "SR-PVX: A source routing based network virtualization hypervisor to enable POF-FIS programmability in vSDNs," *IEEE Access*, vol. 5, pp. 7659–7666, 2017.

[42] M. Garey and D. Johnson, *Computers and Intractability: a Guide to the Theory of NP-Completeness*. W. H. Freeman & Co. New York, 1979.

[43] D. Castro-Silva and E. Gourdin, "A study on load-balanced variants of the bin packing problem," *arXiv preprint arXiv:1810.12086*, 2018. [Online]. Available: https://arxiv.org/abs/1810.12086

[44] G. Strang, "Karmarkar's algorithm and its place in applied mathematics," *Math. Intell.*, vol. 9, pp. 4–10, Jan. 1987.

[45] D. Dubhashi and A. Panconesi, *Concentration of measure for the analysis of randomized algorithms*. Cambridge University Press, 2009.

[46] H. Kuhn, "The Hungarian method for the assignment problem," *Naval Res. Logist. Quart.*, vol. 2, pp. 83–97, Mar. 1955.

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

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

[49] Y. Zhang and N. Ansari, "On architecture design, congestion notification, TCP incast and power consumption in data centers," *IEEE Commun. Surveys Tuts.*, vol. 15, pp. 39–64, First Quarter 2012.

[50] 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