Near-Optimal Deployment of Service Chains by Exploiting Correlations Between Network Functions

Huawei Huang, *Member, IEEE*, Peng Li, *Member, IEEE*, Song Guo, *Senior Member, IEEE*, Weifa Liang, *Senior Member, IEEE*, and Kun Wang, *Senior Member, IEEE*

Abstract—A modern Network Function Virtualization (NFV) service is usually expressed in a service chain that contains a list of ordered network functions, each can run in one or multiple virtual machines. Although lots of efforts have been devoted to service chain deployment, the researchers normally consider a simple model of network functions where different service chains have their own network functions no matter whether some of the network function appliances are interdependent. In this paper, we study the service chain deployment by exploiting two types of correlations between network functions: the *Coordination Effect* due to information exchanges among multiple VMs running the same network function, and the *Traffic-Change Effect* where the volume of outgoing traffic is not necessarily equal to the volume of its incoming traffic at each network function because of packet manipulations such as compression and encryption. These two effects have not been studied simultaneously in the context of service chaining. With theobjective to maximize the profit measured by the admitted traffic minus the implementation cost, we first formulate a joint **s**ervice-f**u**nction de**p**loym**e**nt and **t**raffic scheduling (SUPER) problem that is proved to be NP-hard. We then devise an approximation algorithm based on the Markov approximation technique and analyze its theoretical bound on the convergence time. Simulation results show that the proposed algorithm outperforms two existing benchmark algorithms significantly.

Index Terms—NFV, service chain, coordination effect, traffic-change effect, markov approximation

1 Introduction

Traditionally, network functions such as firewalls, unified threat management, and deep packet inspection are implemented in dedicated hardware appliances. NFV offers a paradigm enabling network functions to be decoupled from hardware and executed as software-based virtualized appliances in virtual machines (VMs), which are hosted on off-the-shelf physical machines (PMs) in datacenters. Thus, NFV offers great flexibility in realizing network services, such as network resilience, service assurance, test/diagnostics and security surveillance. Inspired by the great advantages of NFV, various design of programmable middleboxes has been proposed recently [1], [2], [3]. Combining with the software-defined networking (SDN) [4], NFV significantly simplifies the management of network services [5], [6], [7], [8].

The promises of NFV stemming from its independence from hardware also pose a new challenge of fully exploiting the flexibility of optimizing network function deployments. In practice, a modern network service is usually expressed in a service chain that is composed of a sequence of network functions with each running in one or multiple VMs. One such a service chain example with three network functions: firewall, intrusion detection system (IDS), and video transcoder, is shown in Fig. 1, where multiple VMs are launched for each network function. Network flows should go through these network functions in order to accomplish the service.

Although many existing studies [7], [9], [10], [11] are devoting to service chain deployments, they considered a simple model that VMs running the same network function are independent of each other. However, some recently emerged network functions impose *coordinations* among their VMs. For example, in online distributed intrusion detection approaches proposed in [12], [13], multiple distributed servers need to periodically exchange a number of sample data to detect intrusions cooperatively. Such coordinations lead to an overhead that would decrease the VM processing rate as more VMs are launched. In this paper, we call this overhead as the *Coordination Effect*, which is taken into account in our approach while deploying service chains on a cluster of VMs. As an example shown in Fig. 1, when only one VM (B_1 or B_2) is launched for network function B, its processing capability

- *H. Huang is with the Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong, and also University of Aizu, Aizuwakamatsu 965-8580, Japan. E-mail: cshwhuang@comp.polyu.edu.hk.*
- *S. Guo is with the Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong. E-mail: song.guo@polyu.edu.hk.*
- *P. Li is with the University of Aizu, Aizuwakamatsu 965-8580, Japan. E-mail: pengli@u-aizu.ac.jp.*
- *W. Liang is with the Research School of Computer Science, Australian National Universit, Canberr, ACT 0200, Australia. E-mail: wliang@cs.anu.edu.au.*
- *K. Wang is with the Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu 210093, China. E-mail: kwang@njupt.edu.cn.*

Manuscript received 4 Mar. 2017; revised 15 Sept. 2017; accepted 1 Dec. 2017. Date of publication 6 Dec. 2017; date of current version 9 June 2020. (Corresponding author: Song Guo.) Recommended for acceptance by J. Li. Digital Object Identifier no. 10.1109/TCC.2017.2780165

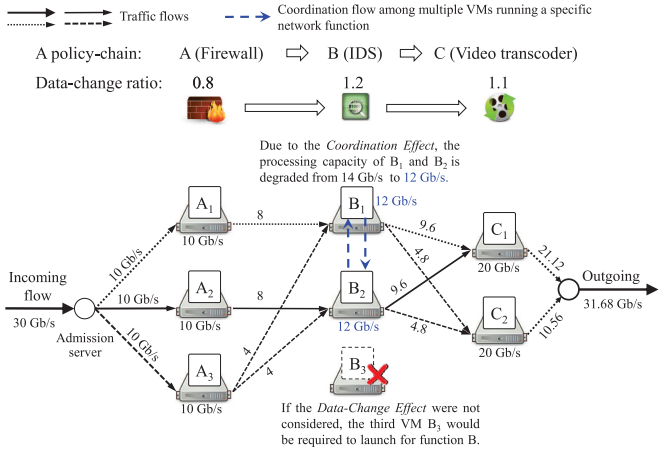


Fig. 1. An example to illustrate both *Coordination Effect* and *Traffic-Change Effect*, while deploying service chains in a cloud network. The *Coordination Effect* refers to the degradation of service capability due to data exchange among the multiple VM instances launched for a specific network function. Then, the *Traffic-Change ratio* refers to the ratio of traffic-changing in a network function between the outgoing and the incoming volumes.

is 14 Gb/s. Due to the impact of *Coordination Effect*, when launching the second VM for B, the data processing capabilities of both B_1 and B_2 are reduced from the original 14 Gb/s to the current 12 Gb/s.

To the best of our knowledge, we are the first to consider the *Coordination Effect* among multiple VMs launched for a specific network function when conducting service chaining. The significant impact on the problem-complexity of such effect will be further discussed in the proof of Theorem 1.

On the other hand, unlike switches and routers that are used to interconnect networks, we notice that some network functions for inspecting and manipulating traffic potentially change the volume of incoming traffic. For example, a VPN proxy enlarges traffic rates because of the Internet Protocol Security (IPsec) header overhead [14], a video transcoder can change packet size by converting a packet format from one type to another type, and a firewall drops packets that violate predefined security policies. We here term this effect as the *Traffic-Change Effect* in terms of traffic-changing in a network function. The ratio of traffic-changing in a network function of the outgoing volume to its incoming volume is called the *Traffic-Change ratio*. The example in Fig. 1 also illustrates the significant impact of *Traffic-Change Effect* on the service chain deployment. As shown, a network flow with an arriving rate of 30 Gb/s requests a service chain consisting of three network functions A, B and C. Their corresponding VMs can process the traffic at rates of 10 Gb/s, 14 Gb/s, and 20 Gb/s, respectively. To ensure that each function can serve with a rate of 30 Gb/s, the traditional deployment scheme that ignored the correlations between network functions will deploy 3, 3, and 2 VMs for functions A, B and C, respectively to meet the resource demands. However, due to the *Traffic-Change Effect*, the amount of traffic processed by these functions are different. For example, three VMs of function A are launched to process the incoming traffic at a rate of 30 Gb/s. After filtering out some packets by the firewall, the total amount of traffic forwarded to function B is 24 Gb/s. Therefore, the traffic flow injected to function B can be served by using only 2 VMs B_1 and B_2 even though their processing rate has been decreased to 12 Gb/s. As a matter of fact, *Traffic-Change Effect* was mentioned in [15], [16], but has been largely ignored when deploying service chains in cloud datacenter networks.

Furthermore, service chain deployment becomes more challenging when multiple network flows request different NFV services simultaneously. For a given number of requested network functions, we need to determine the number of VMs deployed for each network function and on which PMs they should be launched. From the point of view of NFV service provider, although launching more VMs can increase NFV service capability by admitting more traffic, this results in a higher cost of resource utilization. To exploit a fine tradeoff between service capability and resource utilization for service providers, we study a joint **S**ervice-f**U**nction de**P**loym**E**nt and t**R**affic scheduling (SUPER) problem with the objective to maximize the profit, which is the total revenue collected by admitted traffic flow minus the cost of launched VMs. To efficiently address this problem, we propose an efficient algorithm based on Markov approximation with performance guarantee in this paper.

The main contributions of this paper are summarized as follows.

- We study the service chain deployment by considering two types of correlations jointly: *Coordination Effect* among multiple VMs launched for a *same* network function; and *Traffic-Change Effect* between *different* network functions.
- By jointly considering VM deployment and traffic scheduling, we formulate the defined SUPER problem as a mixed integer linear programming for the problem. The NP-hardness of this problem is also proved.
- We then devise an efficient approximation algorithm based on the Markov approximation technique. The theoretical bound on the convergence time of this proposed algorithm is derived.
- Simulation results demonstrate that the proposed algorithm outperforms state-of-the-art benchmarks in terms of both the network profit and the convergence speed.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 elaborates the system model and defines the problem precisely. The Markov approximation based algorithm is presented in Section 4. Section 5 conducts the performance evaluations. Finally, Section 6 concludes the paper.

2 Related Work

Recent research efforts on network service chain can be classified into three categories. First, several studies [2], [3], [10] focused on the optimal traffic routing, under a given number of service chains. For example, Cao et al. [10] studied a routing steering problem for a set of policy-specific flows that need to pass through a logical sequence of network functions in SDN networks.

Second, optimal placements of network functions have been explored in [17], [18], [19], [20], [21]. For instance, Liu et al. [19] focused on the optimal placement of middleboxes to minimize the end-to-end delay and bandwidth

occupation in service chains. To reduce the expensive optical/electronic/optical conversions for the packet/optical datacenters, Xia et al. [20] proposed a heuristic algorithm to efficiently place network functions of each service chain into fewer pods.

Finally, the other group of studies (e.g., [5], [11], [16], [22]) jointly considered the deployment of computing and network resources. Particularly, the placement of network functions and traffic routing is jointly optimized. For example, Gember et al. [5] implemented an orchestration layer for virtual middleboxes by using the SDN technique. The proposed systematic tool Stratos can provide orchestrations in separated three steps: horizontal scaling, middlebox placement and traffic engineering. Kuo et al. [11] recently studied a joint optimization problem of network function placement and routing path selection to maximize the resource occupation. They proposed a dynamic programming algorithm, which handles demands sequentially. Then, Li et al. [22] designed a system named NFV-RT that can dynamically allocate the resources in NFV networks, aiming to maximize the number of admitted requests. Gu et al. [16] presented a market mechanism design in terms of dynamic pricing and provisioning of service chains in a datacenter. By applying an efficient auction mechanism, they formulate the NFV resource allocation as a social welfare maximization problem.

In contrast to the mentioned studies, our work is in the third category, we study a service chain deployment problem by exploiting two types of correlations between network functions simultaneously: the *Coordination Effect* among multiple VMs for the same specific network function, and the *Traffic-Change Effect* between different network functions. We also notice that [23], [24], [25] are the most related studies to our work. The ingress/egress bit-rate variations at VNFs mentioned in [23] and the compression/decompression factor of flows considered in [24] are in fact Traffic-Change effects. In addition, [25] mentions state-sharing among the replicas of a network function. However, all the three studies do not study the *negative effect* caused by the coordinations among multiple replicas as we do.

3 Problem Statement and Formulation

3.1 System Model

We consider an NFV cluster that is composed of a set $\mathbb{P}$ of Physical Machines (PMs) that contain virtual machines to provide NFV services for incoming flows. Thanks to the advance of networking technology, these machines can be connected via a well-connected network. Since some network functions are compute-intensive, NFV services are usually constrained by computation resource instead of network transmission. The hardware resource constraint on each physical machine $p \in \mathbb{P}$ is denoted by E_p.

There are a set $\mathbb{D}$ of traffic flows and each flow $d \in \mathbb{D}$ with rate A^d requests NFV services in the form of a service chain $\mathbb{U}^d = \{u_1, u_2, \ldots\}$, where $u \in \mathbb{U}^d$ denotes a specific network function. A network function $u \in \mathbb{U}^d$ can be implemented as a set V_u^d of VMs, and each VM $v \in V_u^d$ can be deployed at one physical machine $p \in \mathbb{P}$ with the amount of e_v resource occupation. Since network functions provide service sequentially, we use $H_{prev}(v)$ and $H_{next}(v)$ to denote the sets of VMs in v's previous and next hops, respectively.

TABLE 1
Symbols and Variables

Notation	Description
$\mathbb{P}$	the set of Physical Machines (PMs)
$\mathbb{D}$	a set of given incoming network flows
$\mathbb{U}^d$	the policy chain for flow $d \in \mathbb{D}$, e.g., one policy chain is shown as $\{u_1, u_2, u_3\}$.
A^d	traffic rate of arriving flow $d \in \mathbb{D}$
V_u^d	the set of VMs launched to run network function $u \in \mathbb{U}^d$
$C_u^d(.)$	non-increasing function, which returns the data processing capability of VM v ($v \in V_u^d$), due to the *Coordination Effect*
e_v	unit cost of bottleneck-resource (such as CPU capability) in PMs to deploy VM v
E_p	the capacity of bottleneck-resource in PM $p \in \mathbb{P}$, hence a PM can support only a limited num. of VMs
π_v	the *Traffic-Change ratio* on VM $v \in V_u^d$
$H_{prev}(v)$,	the set of VMs for the network function in
$H_{next}(v)$	the previous/next hop of $u \in \mathbb{U}^d$ ($v \in V_u^d$)
$x_{v,p}^d$	binary variable indicating whether VM $v \in V_u^d$ ($u \in \mathbb{U}^d$) is deployed at PM $p \in \mathbb{P}$
$f_{v,v'}^d$	continuous variable denoting the non-negative traffic rate passing from VM $v \in V_u^d$ to VM $v' \in H_{next}(v)$ for flow $d \in \mathbb{D}$
$\mathbf{x}$	$= \{x_{v,p}^d, \forall v \in V_u^d, u \in \mathbb{U}^d, d \in \mathbb{D}, p \in \mathbb{P}\}$, the *VM deployment* solution for all traffic flows

For example, given a service chain $\{u_1, u_2, u_3\}$ and VM $v \in V_{u_2}^d$, we have $H_{prev}(v) = V_{u_1}^d$ and $H_{next}(v) = V_{u_3}^d$. In particular, the data processing capability of a VM $v \in V_u^d$ is returned by $C_u^d(.)$, which is a customized non-increasing function over the number of VMs launched for a specific network function u for flow d. The *Traffic-Change ratio* of a VM v is denoted by π_v. Before being admitted to enter the NFV cluster, all flows need to go through an *admission server* denoted by v_0 that determines the traffic rate of each flow. The notations used in this paper are summarized in Table 1.

3.2 Problem Definition

3.2.1 Definition of Variables

We first define a binary variable $x_{v,p}^d$ to denote the VM deployment for network function $u \in \mathbb{U}^d$ as follows.

$$x_{v,p}^d = \begin{cases} 1, & \text{if VM } v \in V_u^d \text{ is launched at PM } p \in \mathbb{P}; \\ 0, & \text{otherwise.} \end{cases}$$

We then define variable $f_{v,v'}^d \in \mathbb{R}_{\geq 0}$ to denote the traffic rate from a VM v to its next hop $v' \in H_{next}(v)$.

3.2.2 Problem Constraints

Since each VM will be deployed to one physical machine only if launched, we have the following constraint

$$\sum_{p \in \mathbb{P}} x_{v,p}^d \leq 1, \forall v \in V_u^d, \forall u \in \mathbb{U}^d, \forall d \in \mathbb{D}. \quad (1)$$

Each physical machine can accommodate a limited number of VMs due to the constraint of its bottleneck-resource, such as CPU, memory and storage. Here we only consider one type of bottleneck resource for all VMs. The formulation considering the constraints of multiple types of bottleneck resource can be easily extended. Thus, the resource related constraint can be written as

$$\sum_{d\in\mathbb{D}}\sum_{u\in\mathbb{U}^d}\sum_{v\in V_u^d} x_{v,p}^d \cdot e_v \le E_p, \forall p \in \mathbb{P}. \tag{2}$$

If a VM v has been deployed to a physical machine, the volume of incoming traffic cannot exceed its data processing capability. Thus, we have the following constraint

$$\sum_{v'\in H_{prev}(v)} f_{v',v}^d \le \sum_{p\in\mathbb{P}} x_{v,p}^d \cdot C_u^d(\zeta), \quad \forall v \in V_u^d, u \in \mathbb{U}^d, \forall d \in \mathbb{D}, \tag{3}$$

where $\zeta = \sum_{p\in\mathbb{P}}\sum_{\bar{v}\in V_u^d} x_{\bar{v},p}^d$ is the number of launched VMs running network function u for flow d. Thus, $C_u^d(\zeta)$ returns the data processing capability of each of the multiple VMs $\bar{v} \in V_u^d$, due to the *Coordination Effect*. We can see that the value of function $C_u^d(.)$ depends on the assignment of variables $x_{\bar{v},p}^d(\bar{v} \in V_u^d, p \in \mathbb{P})$.

Different from traditional switches with only data forwarding, VMs with a specific functionality can modify the contents of incoming packets, and the outgoing traffic volume of a flow may not be equal to its incoming volume. This is exactly the aforementioned *Traffic-Change Effect*, which leads to the following flow-conservation constraint

$$\sum_{v'\in H_{prev}(v)} f_{v',v}^d = \pi_v \cdot \sum_{v''\in H_{next}(v)} f_{v,v''}^d, \quad \forall v \in V_u^d, \forall u \in \mathbb{U}^d, \forall d \in \mathbb{D}, \tag{4}$$

where π_v represents the Traffic-Change ratio that describes the traffic scaling factor on VM v.

Furthermore, for each flow $d \in \mathbb{D}$, the amount of admitted traffic can be calculated by

$$\sum_{v'\in H_{next}(v_0)} f_{v_0,v'}^d = R^d, \forall d \in \mathbb{D}, \tag{5}$$

where v_0 denotes the *admission server*, which can be viewed as the data-source for all flows. Also, the accumulative amount of admitted traffic cannot exceed the maximum arriving rate of flow d:

$$R^d \le A^d, \forall d \in \mathbb{D}. \tag{6}$$

3.2.3 Optimization Objective

On one hand, we need to maximize the total amount of admitted traffic (represented by Γ) that can be expressed by

$$\Gamma = \sum_{d\in\mathbb{D}} R^d. \tag{7}$$

On the other hand, maximizing Γ requires increasing the traffic processing rate by launching more VMs. This would lead to a higher cost, denoted by Ψ, in terms of resource occupations, which is calculated by

$$\Psi = \sum_{d\in\mathbb{D}}\sum_{u\in\mathbb{U}^d}\sum_{v\in V_u^d}\sum_{p\in\mathbb{P}} x_{v,p}^d \cdot e_v, \tag{8}$$

where e_v is the unit cost of the bottleneck resource to deploying the VM v measured by the bottleneck resource consumed. Note that, tuning the value of e_v varies the weight of Ψ in the overall objective. To strive for a better tradeoff between service capability and resource occupation, we define the net profit as $\Lambda = \Gamma - \Psi$, and we formulate the SUPER problem as a Mixed Integer Linear Programming (MILP).

$$\begin{aligned} \text{SUPER}: \max\ & \Lambda = \Gamma - \Psi \\ \text{s.t.}\quad & \text{Constraints (1) to (6)}, \\ & x_{v,p}^d \in \{0,1\},\ f_{v,v'}^d \ge 0, \forall v \in V_u^d, u \in \mathbb{U}^d, \forall d \in \mathbb{D}. \end{aligned}$$

3.2.4 NP-Hardness of the Problem

Theorem 1. *The SUPER problem is NP-hard.*

Proof. We prove the SUPER is NP-hard problem by reducing from the well-known Bounded Knapsack Problem (BKP), which is defined as follows. Given a knapsack with the capacity W, and a set $\mathbb{N}$ of item types, each type of item is with an identical weight w_i and an identical value α_i. Each type $i \in \mathbb{N}$ has c_i numbers of items. The problem is to maximize the total value of selected items to be placed into the knapsack, such that the total weight of items is no greater than the capacity of the knapsack.

$$\begin{aligned} \text{BKP}: \max\ & \sum_{i\in\mathbb{N}} \alpha_i \cdot x_i \\ \text{s.t.}\ & \sum_{i\in\mathbb{N}} w_i \cdot x_i \le W \text{ and } x_i \in \{0,1,\ldots,c_i\}, \end{aligned}$$

where x_i represents the number of type-i items.

We now consider a special case of the SUPER problem where a single physical machine $\hat{p}$ serves a set $\mathbb{D}$ of flows requesting a single network function $\hat{u}$. We further assume that each VM $v \in V_{\hat{u}}^d$ consumes the identical PM bottleneck-resource e_v. With respect to the bottleneck resource constraint, the SUPER formulation in such a special case has only one constraint, i.e., the inequality (2), which can be rewritten as

$$\sum_{d\in\mathbb{D}} e_v \cdot \hat{y}_d \le E_{\hat{p}}, \tag{9}$$

where $\hat{y}_d = \sum_{v\in V_{\hat{u}}^d} x_{v,\hat{p}}^d$ denotes the number of launched VMs running $\hat{u}$ for flow d, and $\hat{y}_d \in \{0,1,\ldots,|V_{\hat{u}}^d|\}$, $\forall d \in \mathbb{D}$.

Next, to calculate the admitted traffic rate R^d for each flow $d \in \mathbb{D}$, we still use the admission server node v_0 to denote the source node for all the admitted flows, which pass through the VMs for the only network function $\hat{u}$. Then, by Equation (5), we have

$$R^d = \sum_{v\in V_{\hat{u}}^d} f_{v_0,v}^d, \forall d \in \mathbb{D}. \tag{10}$$

On the other hand, the objective function of SUPER is rewritten as

$$\begin{aligned}\Lambda &= \Gamma - \Psi \\ &= \sum_{d\in\mathbb{D}} R^d - \sum_{d\in\mathbb{D}}\sum_{v\in V_{\hat{u}}^d} x_{v,\hat{p}}^d \cdot e_v, \\ &= \sum_{d\in\mathbb{D}}\sum_{v\in V_{\hat{u}}^d} (f_{v_0,v}^d - x_{v,\hat{p}}^d \cdot e_v).\end{aligned} \tag{11}$$

By constraint (3), we know that

$$f_{v_0,v}^d \le x_{v,\hat{p}}^d \cdot C_{\hat{u}}^d\left(\sum_{\bar{v}\in V_{\hat{u}}^d} x_{\bar{v},\hat{p}}^d\right), \forall v \in V_{\hat{u}}^d, \forall d \in \mathbb{D}. \tag{12}$$

We let $f_{v_0,v}^d = \xi_d \cdot x_{v,\hat{p}}^d \cdot C_{\hat{u}}^d(\sum_{\bar{v}\in V_{\hat{u}}^d} x_{\bar{v},\hat{p}}^d), \forall v \in V_{\hat{u}}^d, \forall d \in \mathbb{D}$, where the auxiliary constant coefficient $\xi_d \in [0,1]$. Then, Equation (11) can be rewritten as

$$\Lambda = \sum_{d\in\mathbb{D}}\sum_{v\in V_{\hat{u}}^d} x_{v,\hat{p}}^d \left[\xi_d \cdot C_{\hat{u}}^d\left(\sum_{\bar{v}\in V_{\hat{u}}^d} x_{\bar{v},\hat{p}}^d\right) - e_v\right]. \tag{13}$$

Finally, letting $\beta_d = \xi_d \cdot C_{\hat{u}}^d(\sum_{\bar{v}\in V_{\hat{u}}^d} x_{\bar{v},\hat{p}}^d) - e_v$, the objective of the SUPER problem in Equation (11) can be reformulated as

$$\begin{aligned}&\max \sum_{d\in\mathbb{D}} \beta_d \cdot \hat{y}_d \\ &\text{s.t. } (9), \text{ and } \hat{y}_d \in \{0,1,\ldots,|V_{\hat{u}}^d|\}, \forall d \in \mathbb{D}.\end{aligned} \tag{14}$$

Discussion: Taking advantage of β_d, we can infer that if the *Coordination Effect* were not considered, the term $C_{\hat{u}}^d(\sum_{\bar{v}\in V_{\hat{u}}^d} x_{\bar{v},\hat{p}}^d)$ would be a constant representing the capability of VM $v \in V_{\hat{u}}^d$, resulting in that β_d becomes a constant, since e_v is a constant as well. Further, if β_d were a constant, then the formulation (14) would become BKP. However, β_d is in fact a function of variable $x_{\bar{v},\hat{p}}^d$ ($\bar{v} \in V_{\hat{u}}^d$), which makes β_d a variable, too. Therefore, formulation (14) is even harder than BKP. Here, we also can see that such *Coordination Effect* further complicates the service chain deployment for the SUPER problem. Due to the NP-hardness of BKP, we can conclude that the SUPER problem is NP-hard, too. □

4 Near-Optimal Markov-Chain Based Algorithm

A number of approaches [26], [27], [28] have been proposed to solve the MILP problems. In this section, we propose a fast approximation algorithm for the SUPER problem based on the Markov approximation technique [29] that has been adopted by several works [30], [31], [32], [33], [34]. Algorithm based on the Markov approximation technique will include the theory of Log-Sum-Exp approximation and the Markov chain modeling. It should be noticed that the state definition in the Markov chain, the transition rate between different states, and the detailed operations behind each transition need to be devised creatively for different optimization problems.

Based on the Markov approximation technique [29], one can build a framework to solving the combinatorial optimization, where the global optimal solution consists of distributed decisions on each component of a system. The insight to adopt this technique in this paper is that the SUPER problem is in fact a combinatorial optimization. Applying the Markov approximation, we can find a near-optimal configuration for the holistic system.

4.1 Approximation Algorithm

4.1.1 Log-Sum-Exp Approximation

By carefully examining the SUPER formulation, we find that once the values of variables $x_{v,p}^d$ are fixed, it becomes a linear programming that can be easily solved. Letting $\mathbf{x} = \{x_{v,p}^d, \forall v \in V_u^d, \forall u \in \mathbb{U}^d, \forall d \in \mathbb{D}, \forall p \in \mathbb{P}\}$ denote a feasible deployment of VMs, we have the SUPER-LP problem shown as follows:

$$\begin{aligned}&\text{SUPER-LP}(\mathbf{x}) : \max \hat{\Lambda} = \Gamma(\mathbf{x}) - \Psi(\mathbf{x}) \\ &\text{s.t. } (3), (4) \text{ and } (6), \\ &f_{v,v'}^d \ge 0, \ \ \forall v \in V_u^d, v' \in H_{next}(v), \forall d \in \mathbb{D}.\end{aligned}$$

We define the set $\mathcal{X}$ to represent the space of all possible solutions to SUPER problem. Then, all the feasible solutions to SUPER-LP problems are members of $\mathcal{X}$. Let $\hat{\Lambda}_{\mathbf{x}}$ denote the maximum profit of SUPER-LP under a given $\mathbf{x}$, the SUPER problem can be approximated by SUPER-MA as follows:

$$\begin{aligned}\text{SUPER-MA} : \max &\sum_{\mathbf{x}\in\mathcal{X}} p_{\mathbf{x}}\hat{\Lambda}_{\mathbf{x}} - \frac{1}{\beta}\sum_{\mathbf{x}\in\mathcal{X}} p_{\mathbf{x}} \log\ p_{\mathbf{x}} \\ \text{s.t.:} &\sum_{\mathbf{x}\in\mathcal{X}} p_{\mathbf{x}} = 1,\end{aligned} \tag{15}$$

where $p_{\mathbf{x}}$ is the probability of using the VM deployment $\mathbf{x}$, and β is a positive constant that controls the approximation accuracy. It is easy to see that the approximation gap between SUPER and SUPER-MA approaches zero when β approaches infinity.

According to the Karush-Kuhn-Tucker (KKT) conditions [35], the optimal solution of the SUPER-MA problem is

$$p_{\mathbf{x}}^* = \frac{\exp(\beta\hat{\Lambda}_{\mathbf{x}})}{\sum_{\mathbf{x}'\in\mathcal{X}}\exp(\beta\hat{\Lambda}_{\mathbf{x}'})}, \forall \mathbf{x} \in \mathcal{X}. \tag{16}$$

If we deploy different configurations in a time-sharing manner, according to the optimal solution $p_{\mathbf{x}}^*$ in Equation (16), then SUPER problem can be solved approximately with an optimality-loss $\frac{1}{\beta}\log|\mathcal{X}|$.

4.1.2 Markov Chain Design

We then design a Markov chain by defining a state space and transition rates so that an approximate solution can be obtained if the Markov chain converges in the end.

We now define the state space of solutions. The transition rate $q_{\mathbf{x},\mathbf{x}'}$ between any two states $\mathbf{x}$ and $\mathbf{x}'$ can be calculated by

$$q_{\mathbf{x},\mathbf{x}'} = \exp\left(\frac{1}{2}\beta(\hat{\Lambda}_{\mathbf{x}'} - \hat{\Lambda}_{\mathbf{x}}) - \tau\right), \tag{17}$$

where τ is a non-negative constant. The transition rate $q_{\mathbf{x},\mathbf{x}'}$ increases with the growth of performance gap between two states.

Algorithm 1. MC Based Algorithm to Solve SUPER

Input: $\mathbb{D}, \mathbb{P}$
1 /* Initialization */
2 initialize all variables in $\mathbf{x}$ to be zero
3 $\overline{V}_u^d = \emptyset$
4 **for** $\forall u \in \mathbb{U}^d, \forall d \in \mathbb{D}$ **do**
5 $x_{\overline{v},p}^d \leftarrow 1$, where $\overline{v} \in V_u^d$ and $p \in \mathbb{P}$ is a physical machine that can accommodate $\overline{v}$
6 $V_u^d = V_u^d - \{\overline{v}\}, \overline{V}_u^d = \overline{V}_u^d \cup \{\overline{v}\}$
7 update $\mathbf{x}$
8 **while** *the solution does not converge* **do**
9 /* setup timers for all flows */
10 **for** $\forall d \in \mathbb{D}$ **do**
11 $\{T_d^+, T_d^-, \Omega_d, \Pi_d\}$ = SetTimer$(d, \mathbf{x})$
12 listen to any timer's expiration
13 /* state transition */
14 **if** *any* T_d^+ *expires* **then**
15 **for** $\forall (u, v, p) \in \Omega_d$ **do**
16 $x_{v,p}^d \leftarrow 1$
17 $V_u^d = V_u^d - \{v\}, \overline{V}_u^d = \overline{V}_u^d \cup \{v\}$
18 update $\mathbf{x}$
19 **if** *any* T_d^- *expires* **then**
20 **for** $\forall (u, v, p) \in \Pi_d$ **do**
21 $x_{v,p}^d \leftarrow 0$
22 $V_u^d = V_u^d + \{\overline{v}\}, \overline{V}_u^d = \overline{V}_u^d - \{v\}$
23 update $\mathbf{x}$

4.2 Algorithm Design

In the following we devise a distributed algorithm to implement the Markov chain with the transition rate (17). The basic idea is to randomly initialize a feasible deployment of all flows, and then periodically update the deployment according to a sophisticated random process, until the solution to SUPER problem converges. Note that, the computing tasks for all flows can be amortized by multiple controllers, since the proposed approach is essentially executed in a distributed manner only with a few synchronization operations.

In particular, the proposed approach includes a main algorithm (Algorithm 1) and a supporting function (Algorithm 2). In the following, we first give the main algorithm design, and then elaborate how to set random timers for deployment updates, which is the key to accelerate the convergence of this approach.

4.2.1 Main Algorithm

All variables in $\mathbf{x}$ to be zero initially. For each network function requested by flows, we launch one VM on one physical machine randomly picked with enough resources to accommodate the VM, as shown in lines 4-5 of Algorithm 1. Note that when $\mathbf{x}$ is fixed, a solution can obtained by solving a linear programming SUPER-LP($\mathbf{x}$). Once an VM $\overline{v}$ is deployed to a physical machine, it will be removed from set V_u^d, immediately.

In the *while* loop between lines 8 and 23 of Algorithm 1, we iteratively update the deployment until the solution converges. In each iteration, we set timers for each flow by invoking function SetTimer$(d, \mathbf{x})$ whose design will be presented in Algorithm 2. This function returns T_d^+ and T_d^-, indicating the timers of adding and removing an VM chain, respectively. Also, the candidate VM chains for adding and removing are maintained in the returned sets Ω_d and Π_d (shown in line 11), respectively. After all timers are set, we start to count down and listen to any timer's expiration.

Algorithm 2. SetTimer$(d, \mathbf{x})$

1 $\mathbf{x}' \leftarrow \emptyset$
2 $\Omega_d \leftarrow \emptyset$ /*an empty chain for d*/
3 **for** $\forall u \in \mathbb{U}^d$ **do**
4 $p \leftarrow$ one feasible PM randomly selected from $\mathbb{P}$
5 $v \leftarrow$ an VM in V_u^d
6 $\Omega_d \leftarrow \Omega_d \cup \{(u, v, p)\}$
7 **if** $\Omega_d \neq \emptyset$ **then**
8 **for** $\forall (u, v, p) \in \Omega_d$ **do**
9 $\mathbf{x}' \leftarrow \mathbf{x}' \cup \{x_{v,p}^d\}$
10 $\triangle \leftarrow \sum_{d \in \mathbb{D}} \prod_{i=1}^{|\mathbb{U}^d|} \sigma_i^d$
11 $T_d^+ = \frac{1}{\triangle} \exp(\tau + \frac{1}{2}\beta(\Lambda_{\mathbf{x}} - \Lambda_{\mathbf{x}'}))$
12 $\varsigma_d \leftarrow$ currently in-use chains by traffic flow d
13 **if** $|\varsigma_d| \geq 1$ **then**
14 $\Pi_d \leftarrow$ a random chain from ς_d
15 **for** $\forall (u, v, p) \in \Pi_d$ **do**
16 $\mathbf{x}' \leftarrow \mathbf{x}' \setminus \{x_{v,p}^d\}$
17 $\nabla \leftarrow \sum_{d \in \mathbb{D}} \varrho_d$
18 $T_d^- = \frac{1}{\nabla} \exp(\tau + \frac{1}{2}\beta(\Lambda_{\mathbf{x}'} - \Lambda_{\mathbf{x}}))$
19 return T_d^+, T_d^-, Ω_d, and Π_d

If timer T_d^+ expires, we deploy the VMs contained in candidate set Ω_d and update sets V_u^d and $\overline{V}_u^d$, as shown in lines 14-18 of Algorithm 1. Similarly, the VMs in set Π_d are removed if any timer T_d^- expires first. This iteration terminates when the deployment $\mathbf{x}$ is updated.

4.2.2 Timer Design

The pseudo code of function SetTimer is shown in Algorithm 2. We first find a set of candidate VMs that can be deployed for performance enhancement. If such VMs can be found, we create a new deployment supposing they are launched on physical machines, and calculate the timer T_d^+, as shown in line 11 of Algorithm 2. Denoted by $\triangle$ the number of possible chains for all traffic flows and $\triangle = \sum_{d \in \mathbb{D}} \prod_{i=1}^{|\mathbb{U}^d|} \sigma_i^d$, where σ_i^d represents the number of feasible PMs for the ith network function in $\mathbb{U}^d$.

To calculate another timer T_d^-, we first find a set Π_d of candidate VMs that can be removed from the current deployment. The total number of in-use chains for all traffic flows can be computed as $\nabla = \sum_{d \in \mathbb{D}} \varrho_d$, where ϱ_d is the number of chains that are being used by d. Finally, Algorithm 2 will deliver both timers T_d^+ and T_d^- and their corresponding candidate VM sets Ω_d and Π_d.

For easy understanding, we use an example to illustrate the details of Algorithm 1 on deploying VMs for network flows. As shown in Fig. 2, a network flow d_1 requires a policy chain {A, B, C}, and it has been assigned two chains $A_1 \rightarrow B_1 \rightarrow C_1$ and $A_2 \rightarrow B_2 \rightarrow C_1$ at the current state $\mathbf{x}$. Suppose timer $T_{d_1}^+$ now expires, system transits to another state $\mathbf{x}'$, by adding a new VM chain $A_3 \rightarrow B_1 \rightarrow C_1$ for d_1, and update the timers of all flows. Since this new chain shares the bottleneck VM B_1 with the first chain, its achievable processing rate is only 5 Gb/s. In addition, due to the *Coordination Effect* among the three VMs running function

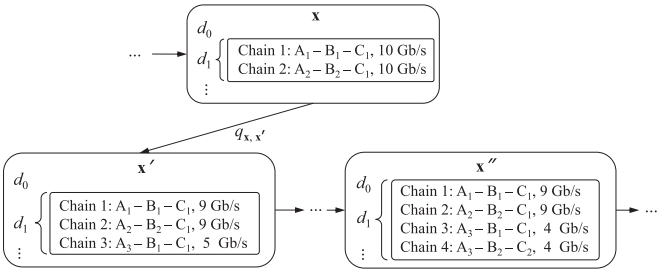


Fig. 2. Example of transitions.

A, the processing rates of both chains 1 and 2 are degraded to 9 Gb/s. The whole system further transits to a state $\mathbf{x}''$ when $T^+_{d_1}$ first expires, in which a new chain $A_3 \rightarrow B_2 \rightarrow C_2$ are launched for d_1. Similarly, due to the coordinations between C_1 and C_2, the processing rate of both Chains 3 and 4 only has 4 Gb/s.

4.3 Algorithm Analysis

We finally analyze the properties of the proposed algorithm. We first prove that the proposed algorithm can realize a time-reversible Markov chain with the stationary distribution shown in (16) through the following three lemmas.

Lemma 1. *All the transition rates of the state-hopping in the devised Markov chain are finite.*

Proof. It can be seen that all transition rates are finite from (17). □

Lemma 2. *The constructed Markov chain is irreducible.*

Proof. According to the definition of state space, all configurations are reachable from each other within a finite number of transitions. Therefore, the constructed Markov chain is irreducible. □

Lemma 3. *The* detailed balance equations *[36] hold in the constructed Markov chain.*

Proof. In Algorithm 2, we set the sojourn time of each configuration to a random variable with an exponential distribution and the transition probability between two configurations is independent. We now analyze the transition rate for adding network service chains.

Let $Pr_{\mathbf{x}\rightarrow\mathbf{x}'}$ denote the transition probability from the current state $\mathbf{x}$ to the next state $\mathbf{x}'$ when a timer T^+_d expires ($\mathbf{x},\mathbf{x}' \in \mathcal{X}$). We define set $S_{\mathbf{x}}(d)(\mathbf{x} \in \mathcal{X})$ to maintain all neighboring states of $\mathbf{x}$. According to Algorithm 2, the current deployment $\mathbf{x}$ can transit to any state $\mathbf{x}' \in S_{\mathbf{x}}(d)$ with the equal probability. When the algorithm selects the next feasible service chain for flow d, there are $\prod_{i=1}^{|\mathbb{U}^d|}\sigma_i^d$ choices in total. We thus have $|S_{\mathbf{x}}(d)| = \prod_{i=1}^{|\mathbb{U}^d|}\sigma_i^d$. The probability $Pr_{\mathbf{x}\rightarrow\mathbf{x}'}$ thus is

$$Pr_{\mathbf{x}\rightarrow\mathbf{x}'} = \frac{1}{|S_{\mathbf{x}}(d)|}, \forall \mathbf{x} \in \mathcal{X}, \forall \mathbf{x}' \in S_{\mathbf{x}}(d). \tag{18}$$

Given the current state $\mathbf{x}$, each timer T^+_d $(\forall d \in \mathbb{D})$ counts down with a rate

$$\rho^+_{\mathbf{x},\mathbf{x}'}(d) = \triangle \cdot \exp^{-1}\left(\tau + \frac{1}{2}\beta(\Lambda_{\mathbf{x}} - \Lambda_{\mathbf{x}'})\right), \tag{19}$$

where τ is constant and the aggregate rate over all $d \in \mathbb{D}$ is

$$\rho_{\mathbf{x},\mathbf{x}'} = \sum_{d\in\mathbb{D}} \rho^+_{\mathbf{x},\mathbf{x}'}(d). \tag{20}$$

The transition rate from $\mathbf{x}$ to $\mathbf{x}'$ is then calculated as follows.

$$\begin{aligned} q_{\mathbf{x},\mathbf{x}'} &= \rho_{\mathbf{x},\mathbf{x}'} \times Pr_{\mathbf{x}\rightarrow\mathbf{x}'} = \sum_{d\in\mathbb{D}} \rho^+_{\mathbf{x},\mathbf{x}'}(d) \times \frac{1}{\prod_{i=1}^{|\mathbb{U}^d|}\sigma_i^d} \\ &= \exp\left(\frac{1}{2}\beta(\Lambda_{\mathbf{x}'} - \Lambda_{\mathbf{x}}) - \tau\right). \end{aligned} \tag{21}$$

On the other hand, let $\rho^-_{\mathbf{x}',\mathbf{x}}(d)$ denote the count-down rate of removing a service chain, which can be expressed by

$$\rho^-_{\mathbf{x}',\mathbf{x}}(d) = \nabla \cdot \exp^{-1}\left(\tau + \frac{1}{2}\beta(\Lambda_{\mathbf{x}'} - \Lambda_{\mathbf{x}})\right). \tag{22}$$

The transition rate $q_{\mathbf{x}',\mathbf{x}}$ can be calculated similarly

$$\begin{aligned} q_{\mathbf{x}',\mathbf{x}} &= \rho_{\mathbf{x}',\mathbf{x}} \times Pr_{\mathbf{x}'\rightarrow\mathbf{x}} = \sum_{d\in\mathbb{D}} \rho^-_{\mathbf{x}',\mathbf{x}}(d) \times \frac{1}{\varrho_d} \\ &= \exp\left(\frac{1}{2}\beta(\Lambda_{\mathbf{x}} - \Lambda_{\mathbf{x}'}) - \tau\right). \end{aligned} \tag{23}$$

Finally, we have $p^*_{\mathbf{x}} q_{\mathbf{x},\mathbf{x}'} = p^*_{\mathbf{x}'} q_{\mathbf{x}',\mathbf{x}}, \forall \mathbf{x},\mathbf{x}' \in \mathcal{X}$ based on (21) (23) and (16). Therefore, the *detailed balance equations* hold in the designed Markov chain. □

From Lemmas 1, 2, and 3, we have the following conclusion according to [36].

Corollary 1. *Algorithm 1 realizes a time-reversible Markov chain with the stationary distribution shown in (16).*

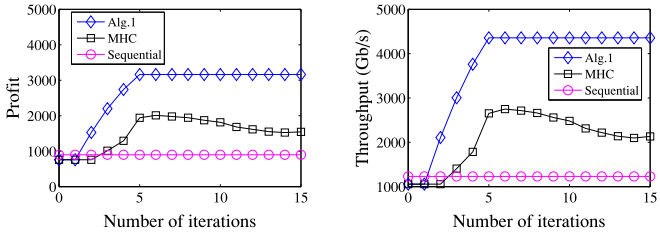
Next, we analyze the *convergence time* of the proposed Algorithm 1. In general, the convergence time of a Markov chain can be described by the *mixing time* [29] of Markovan random field. Let $\boldsymbol{H}_t(\mathbf{x})$ denote the probability distribution space of all states in $\mathcal{X}$ at time t if the initial state is $\mathbf{x}$. Recall that $\boldsymbol{p}^*$ in (16) is the stationary distribution of the constructed Markov chain. We then define its mixing time as follows:

$$t_{mix}(\epsilon) = \inf\{t \geq 0 : \max_{\mathbf{x}\in\mathcal{X}} \|\boldsymbol{H}_t(\mathbf{x}) - \boldsymbol{p}^*\|_{TV} \leq \epsilon\}, \tag{24}$$

where $\epsilon > 0$ determines the performance of convergence and term $\|.\|_{TV}$ denotes the total variance distance between the probability distributions $\boldsymbol{H}_t(\mathbf{x})$ and the optimal one $\boldsymbol{p}^*$. Here the total variance distance can be calculated as $\|\boldsymbol{H}_t(\mathbf{x}) - \boldsymbol{p}^*\|_{TV} = \frac{1}{2}\sum_{x\in\mathcal{X}} |\boldsymbol{H}_t(\mathbf{x}) - p^*(\mathbf{x})|$. We further define $\Lambda_{\max} = \max_{\forall\mathbf{x}\in\mathcal{X}}\Lambda_{\mathbf{x}}$, $\Lambda_{\min} = \min_{\forall\mathbf{x}\in\mathcal{X}}\Lambda_{\mathbf{x}}$, $y = |\mathbb{U}^d|$, and $\kappa = \prod_{d\in\mathbb{D}}(\binom{|\mathbb{P}|}{M_d})^y$. We then conclude on the mixing time by the following Theorem.

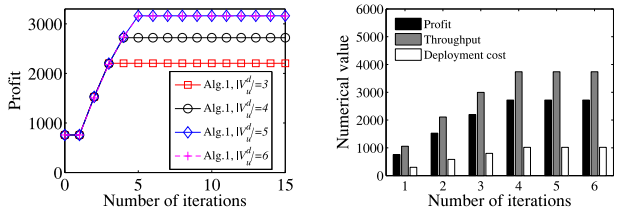
Theorem 2. *The mixing time $t_{mix}(\epsilon)$ for the Markov chain constructed in Algorithm 1 is bounded, i.e.,*

$$t_{mix}(\epsilon) \geq \frac{\exp[\tau - \frac{1}{2}\beta(\Lambda_{\max} - \Lambda_{\min})]}{2\delta} \ln\frac{1}{2\epsilon}, \tag{25}$$



(a) Profit (b) Throughput

Fig. 3. Algorithm 1 shows convergence and overwhelming performance over benchmark algorithms, when $|V_u^d|$=5, $|\mathbb{D}|$=100.



(a) Profit, $|V_u^d| \in \{3, 4, 5, 6\}$ (b) Comparison of 3 metrics, $|V_u^d|$=4

Fig. 4. Performance of Algorithm 1 when $|\mathbb{D}| = 100$.

and

$$t_{mix}(\epsilon) \leq 2\delta\kappa^2 \exp\left[\frac{3}{2}\beta(\Lambda_{\max} - \Lambda_{\min}) + \tau\right] \cdot \left[\ln\frac{1}{2\epsilon} + \frac{1}{2}\ln\kappa + \frac{1}{2}\beta(\Lambda_{\max} - \Lambda_{\min})\right], \tag{26}$$

where the

$$\delta = \begin{cases} \sum_{d\in\mathbb{D}} |\mathbb{P}|^y, & \text{(a) if system tends to add chains;} \\ \sum_{d\in\mathbb{D}} y^{M_d}, & \text{(b) if system tends to remove chains,} \end{cases}$$

and $M_d = \min_{u\in\mathbb{U}^d} |V_u^d|_{(\forall d\in\mathbb{D})}$.

The detailed proof appears in Appendix A.

5 Performance Evaluation

In this section, we first give simulation settings including the network topology, benchmarks and parameter settings. We then present numerical simulation results and explanations of the proposed algorithm.

5.1 Simulation Settings

We implement a simulator in Python to emulate a realistic 4-pod fat-tree topology, which consists of 200 PMs in 8 racks. The default settings of parameters are described as follows unless otherwise specified.

To simulate the resource availability in the real world, the bottleneck resource capacity of each PM (E_p) is randomly generated between 0 and 32, which can be viewed as the available number of CPU-cores equipped in cloud servers. We also randomly generate 500 network flows with traffic rates distributed in [10, 100] Gb/s according to the exponential distribution. Without loss of generality, the *Traffic-Change ratio* (π_v) of each VM v is randomly assigned between 0.8 and 1.2 for both *shrinking* and *enlarging* Traffic-Change effects. The unit consumption of compute resource (e_v) to deploy each VM v is fixed to 1. To construct the individual policy-chain for each flow, we specify five types of network functions, i.e., Firewall, NAT, Load balancer, DPI and Video transcoder. The sequence of network functions in each policy-chain is randomly generated. For each type of network function, at most $|V_u^d|$ numbers of VMs can be launched. The function $C_u^d(.)$ of each VM's data processing capability is specified in each suite of simulation. In addition, since parameters β and τ in Algorithm 1 can be set freely according to different requirements of near-optimal performance [29], here they are set to 10 and 0, respectively. On the other hand, we compare the performance of the proposed algorithm with the other two benchmark algorithms, which are explained as follows.

- Algorithm *MHC* [29], [31]: This algorithm is also based on the Markov approximation technique. Comparing with our approach, the key difference is that the state transition rate in *MHC* is designed as $q_{\mathbf{x},\mathbf{x}'} \propto \exp^{-1}(-\beta\Lambda_{\mathbf{x}'})$. It adopts the configuration with best performance ever traced within a specified iterations as the final solution.
- Algorithm *Sequential* [11]: In this algorithm, demanding flows are served sequentially in a descending order of their arrival rates.

5.2 Simulation Results

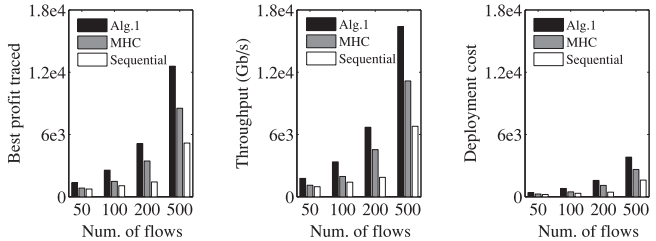
5.2.1 Convergence Property of Algorithm 1

We first study the convergence of the proposed algorithm on dealing with 100 network flows in 15-iteration executions, where each iteration consumes 1 ms logical time. The maximum VM count (i.e., $|V_u^d|$) is fixed at 5. The data processing capability of each VM $v \in V_u^d$ is uniformly and randomly generated in the range of [10, 20] Gb/s. As shown in Fig. 3a, we observe that the proposed Algorithm 1 converges to a solution with a profit that is higher than that by both algorithms *MHC* and *Sequential*. Similar phenomenon can be observed in Fig. 3b which illustrates the throughput (i.e., the total amount of admitted flows) of all algorithms.

We then study the profit performance of the proposed algorithm with different numbers of VMs deployed in Fig. 4a. The profit under the converged solution increases with the growth of numbers of VMs because more flows can be admitted with a larger $|V_u^d|$. However, the profit-improvement becomes saturated of enlarging $|V_u^d|$ from 5 to 6 due to limited compute resources of physical machines. Fig. 4b shows the convergence of all 3 metrics, i.e., the profit, the throughput and the deployment cost, of the proposed algorithm under $|V_u^d|$=4. Although both the throughput and the deployment cost increase when we adjust the deployment, their gap (i.e., profit) finally converges to stable.

5.2.2 Impact of Flow Amount

We then study the impact of the number of flows on the performance of the profit, throughput and deployment cost, by setting $|\mathbb{D}|$ at four different values: 50, 100, 200 and 500, respectively. We have $|V_u^d|$=3 and $|\mathbb{U}^d|$=3. As shown in Fig. 5, we observe that both the throughput and deployment cost increase with the increase on the number of flows. However, throughput improvement brings more profits as shown in Fig. 6a. We attribute this phenomenon to the fact



(a) Maximum profit traced (b) Throughput (c) Deployment cost

Fig. 5. Performance of algorithms when varying the number of flows $|\mathbb{D}| \in${50, 100, 200, 500} under $|V_u^d|$=3, $|\mathbb{U}^d|$ = 3.

that more VMs are launched to serve growing numbers of flows, leading to more traffic admitted.

5.2.3 Impact of Policy-Chain Length

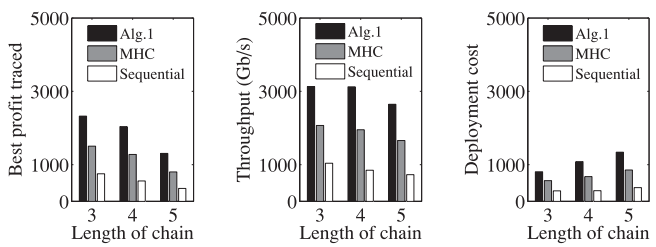
We now vary the length of service chains of flows and show the results in Fig. 6. The number of flows is fixed at 100. We observe that both the profit and the throughput decrease with the growth of the chain length in Figs. 6a and 6b, while the deployment cost is an increasing function under all three algorithms as shown in Fig. 6c. That is because more VMs are launched when the length of service chain becomes longer, leading to a higher deployment cost. However, longer service chains increase lead to higher probability that throughput is constrained by bottleneck VMs with poor processing capability.

5.2.4 Impact of the Coordination Effect

We finally focus on the impact of the *Coordination Effect*. In particular, function $C_u^d(.)$ $(\forall u \in \mathbb{U}^d, \forall d \in \mathbb{D})$ of each VM's data processing capability is set to a linear non-increasing function, the yielded value of which is inversely proportional to the number of VMs launched to run each network function u. In detail, as shown in Fig. 7a, function $C_u^d(.)$ is defined as

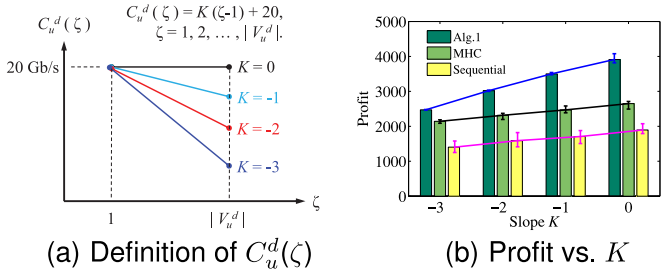
$$C_u^d(\zeta) = K(\zeta - 1) + 20, \ \zeta = 1, 2, \ldots, |V_u^d|, \qquad (27)$$

where *20* (Gb/s) denotes the maximum processing capability for all VMs. As already presented in (3), ζ indicates the total number of VMs launched to run the network function u for flow d. Particularly, as the slope of linear function $C_u^d(\zeta)$, K determines the varying degree of a network function's processing capability when multiple VMs are launched. In this group of simulation, we study the performance of algorithms while varying K within {-3, -2, -1, 0}. The other parameters are set as $|\mathbb{U}^d| = 3$, $|\mathbb{D}| = 100$, and $|V_u^d| = 4$.



(a) Maximum profit traced (b) Throughput (c) Deployment cost

Fig. 6. Performance of algorithms when varying the length of policy-chain (i.e., $|\mathbb{U}^d|$) under $|\mathbb{D}| = 100$, $|V_u^d|$=3.



(a) Definition of $C_u^d(\zeta)$ (b) Profit vs. K

Fig. 7. Performance of algorithms when varying K={-3, -2, -1, 0}.

We evaluate the profit under the converged solution of our proposed Algorithm 1, as well as the best traced solutions of *MHC* and *Sequential* in 15-iteration executions. As shown in Fig. 7b, the profit performance of all algorithms demonstrates a linear growth over the slope K varying from -3 to 0. This is because the processing capability of VMs becomes larger while K increases, leading to a larger profit for all algorithms. In addition, the overwhelming performance of our proposed Algorithm 1 is observed again, comparing with the other two benchmark algorithms.

In summary, the proposed Algorithm 1 can achieve a profit by 30-50 percent and 100-300 percent higher than that by algorithms *MHC* and *Sequential*, respectively. It outperforms the benchmark algorithms in terms of the convergence property as well.

6 Conclusion

In this paper, we studied deploying multiple network service chains for incoming traffic flows on a cluster of virtual machines. To maximize the profit of NFV service providers for serving user flows, we formulated a novel service chain deployment problem, by considering two types of network function correlations: the *Coordination Effect* among VMs running the same network function and the *Traffic-Change Effect* between different network functions. We shown that this problem is NP-hard and devised an approximation algorithm for it, based on the Markov Approximation technique. The convergence time of the proposed algorithm is also analyzed. Evaluation results through simulations show that the proposed algorithm converges fast and outperforms other benchmarks significantly.

Acknowledgments

This work was partially supported by JSPS KAKENHI under Grant Number 16J07062, 16K16038 and the Strategic Information and Communications R&D Promotion Programme (SCOPE No.162302008), MIC, Japan; National Natural Science Foundation of China(61572262); China Postdoctoral Science Foundation (2017T100297 and 2017M610252).

Appendix
Proof of Theorem 2

Proof. Following the framework presented in [37], [38], we show the lower bound and upper bound of mixing time as follows.

We already showed that the constructed continuous-time Markov chain is with a stationary distribution given by Equation (16). When we add new chains, the

minimum probability in the stationary distribution is

$$\begin{aligned} p_{\min} \triangleq \min_{\mathbf{x}\in\mathcal{X}} p^*_{\mathbf{x}} &\geq \frac{\exp(\beta\Lambda_{\min})}{|\mathcal{X}|\exp(\beta\Lambda_{\max})} \\ &\geq \frac{1}{\kappa}\exp(-\beta(\Lambda_{\max}-\Lambda_{\min})) \end{aligned} \tag{28}$$

where $\kappa = \prod_{d\in\mathbb{D}} \binom{|\mathbb{P}|}{M_d}^y$, because $\sum_{\mathbf{x}'\in\mathcal{X}}\exp(\beta\Lambda_{\mathbf{x}'}) \leq |\mathcal{X}|\exp(\beta\Lambda_{\max})$ and $|\mathcal{X}| \leq \kappa$.

We finish this proof by using the *uniformization* technique [29]. Denote by $Q = \{q_{\mathbf{x},\mathbf{x}'}\}$ the transition rate matrix of the Markov chain, we then construct a discrete-time Markov chain $Z(n)$ with a probability transition matrix $P = I + \frac{Q}{\theta}$, where I is the *unit matrix* and θ is the *uniform rate parameter*. We consider a system with successive states from a Markov chain $Z(n)$ and the state of this system at discrete time t is denoted by $Z(N(t))$. This system is an independent Poisson process $N(t)$ with rate θ.

Note that $\sum_{\mathbf{x}\neq\mathbf{x}'} q_{\mathbf{x},\mathbf{x}'} \leq \delta\exp(\frac{1}{2}\beta(\Lambda_{\max}-\Lambda_{\min})-\tau)$ because $q_{\mathbf{x},\mathbf{x}'} \leq \exp(\frac{1}{2}\beta(\Lambda_{\mathbf{x}'}-\Lambda_{\mathbf{x}})-\tau) \leq \exp(\frac{1}{2}\beta(\Lambda_{\max}-\Lambda_{\min})-\tau)$, $\forall \mathbf{x},\mathbf{x}' \in \mathcal{X}$, and $\mathbf{x}$ can transit to at most $\delta = \sum_{d\in\mathbb{D}}|\mathbb{P}|^y$ other states. Therefore, θ is given as

$$\theta = \delta\,\exp\left(\frac{1}{2}\beta(\Lambda_{\max}-\Lambda_{\min})-\tau\right). \tag{29}$$

According to the uniformization theorem [38], the Markov chain and its discrete-time counterpart $Z(N(t))$ has the same distribution. Further, they also share the same stationary distribution (16). Let ρ_2 denote the second largest eigenvalue of transition matrix P for $Z(n)$ and apply the spectral gap inequality [37] and [38], we have

$$\begin{aligned} \frac{\exp(-\theta(1-\rho_2)t)}{2} &\leq \max_{\mathbf{x}\in\mathcal{X}}\|\boldsymbol{H}_t(\mathbf{x})-\boldsymbol{p}^*\|_{TV} \\ &\leq \frac{\exp(-\theta(1-\rho_2)t)}{2(p_{\min})^{\frac{1}{2}}}. \end{aligned}$$

Therefore,

$$\begin{aligned} \frac{1}{\theta(1-\rho_2)}\ln\frac{1}{2\epsilon} &\leq t_{mix}(\epsilon) \\ &\leq \frac{1}{\theta(1-\rho_2)}\left[\ln\frac{1}{2\epsilon}+\frac{1}{2}\ln\frac{1}{p_{\min}}\right]. \end{aligned} \tag{30}$$

Particularly, ρ_2 can be bounded by Cheeger's inequality [37], [38] as

$$1-2\Phi \leq \rho_2 \leq 1-\frac{1}{2}\Phi^2, \tag{31}$$

where Φ is the "conductance" of P, and defined as follows:

$$\Phi \triangleq \min_{N\subset\mathcal{X},\pi_N\in(0,1/2]} \frac{F(N,N^c)}{\pi_N}. \tag{32}$$

Here $\pi_N = \sum_{\mathbf{x}\in N} p^*_{\mathbf{x}}$ and $F(N,N^c) = \sum_{\mathbf{x}\in N,\mathbf{x}'\in N^c} p^*_{\mathbf{x}}P(\mathbf{x},\mathbf{x}')$.

The combination of (30) and (31) yields

$$\frac{1}{2\theta\Phi}\ln\frac{1}{2\epsilon} \leq t_{mix}(\epsilon) \leq \frac{2}{\theta\Phi^2}\left[\ln\frac{1}{2\epsilon}+\frac{1}{2}\ln\frac{1}{p_{\min}}\right]. \tag{33}$$

The upper bound of Φ is then derived for any $N' \subset \mathcal{X}, \pi(N') \in (0,1/2]$, that is,

$$\begin{aligned} \Phi &= \min_{N\subset\mathcal{X},\pi_N\in(0,1/2]} \frac{F(N,N^c)}{\pi_N} \\ &\leq \frac{1}{\pi_{N'}}\sum_{\mathbf{x}\in N,\mathbf{x}'\in N'^c} p^*_{\mathbf{x}}P(\mathbf{x},\mathbf{x}') \\ &= \frac{1}{\pi_{N'}}\sum_{\mathbf{x}\in N'} p^*_{\mathbf{x}}\cdot\left(\sum_{\mathbf{x}'\in N'^c}P(\mathbf{x},\mathbf{x}')\right) \\ &\leq \frac{1}{\pi_{N'}}\sum_{\mathbf{x}\in N'} p^*_{\mathbf{x}} \\ &= 1. \end{aligned} \tag{34}$$

The lower bound of $t_{mix}(\epsilon)$ is then derived by combining (29), (33) and (34), i.e.,

$$\begin{aligned} t_{mix}(\epsilon) &\geq \frac{1}{2\theta}\ln\frac{1}{2\epsilon} \\ &= \frac{\exp[\tau-\frac{1}{2}\beta(\Lambda_{\max}-\Lambda_{\min})]}{2\delta}\ln\frac{1}{2\epsilon}. \end{aligned} \tag{35}$$

Next, we derive the lower bound of Φ. When $q_{\mathbf{x},\mathbf{x}'} \neq 0$, $\forall \mathbf{x},\mathbf{x}' \in \mathcal{X}$, via (17) we know that

$$\begin{aligned} q_{\mathbf{x},\mathbf{x}'} &= \exp\left(\frac{1}{2}\beta(\Lambda_{\mathbf{x}'}-\Lambda_{\mathbf{x}})-\tau\right) \\ &\geq \exp\left(\frac{1}{2}\beta(\Lambda_{\min}-\Lambda_{\max})-\tau\right). \end{aligned} \tag{36}$$

The combination of (32) and (36) outputs

$$\begin{aligned} \Phi &\geq \min_{N\subset\mathcal{X},\pi_N\in(0,1/2]} F(N,N^c) \\ &\geq \min_{\mathbf{x}\neq\mathbf{x}',P(\mathbf{x},\mathbf{x}')>0} F(\mathbf{x},\mathbf{x}') \\ &= \min_{\mathbf{x}\neq\mathbf{x}',P(\mathbf{x},\mathbf{x}')>0} p^*_{\mathbf{x}}P(\mathbf{x},\mathbf{x}') \\ &= \min_{\mathbf{x}\neq\mathbf{x}',P(\mathbf{x},\mathbf{x}')>0} p^*_{\mathbf{x}}\cdot\frac{q_{\mathbf{x},\mathbf{x}'}}{\theta} \\ &\geq \frac{p_{\min}}{\theta}\cdot\exp\left(\frac{1}{2}\beta(\Lambda_{\min}-\Lambda_{\max})-\tau\right). \end{aligned} \tag{37}$$

Finally, combining (33), (28), (29) and (37), we have

$$\begin{aligned} t_{mix}(\epsilon) &\leq \frac{2}{\theta\Phi^2}\left[\ln\frac{1}{2\epsilon}+\frac{1}{2}\ln\frac{1}{p_{\min}}\right] \\ &\leq \frac{2\theta\exp(2\tau-\beta(\Lambda_{\max}-\Lambda_{\min}))}{p^2_{\min}}\left[\ln\frac{1}{2\epsilon}+\frac{1}{2}\ln\frac{1}{p_{\min}}\right] \\ &\leq 2\delta\kappa^2\exp\left[\frac{3}{2}\beta(\Lambda_{\max}-\Lambda_{\min})+\tau\right]\cdot \\ &\quad\left[\ln\frac{1}{2\epsilon}+\frac{1}{2}\ln\kappa+\frac{1}{2}\beta(\Lambda_{\max}-\Lambda_{\min})\right]. \end{aligned} \tag{38}$$

This concludes the proof for case (a). Under case (b), $\forall \mathbf{x},\mathbf{x}' \in \mathcal{X}$, $\mathbf{x}$ may transit to at most $\delta = \sum_{d\in\mathbb{D}} y^{M_d}$ other states. Following the same routine for case (a), the rest of proof for case (b) can be conducted similarly. □

REFERENCES

[1] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, "Design and implementation of a consolidated middlebox architecture," in *Proc. 9th USENIX Conf. Networked Syst. Des. Implementation*, 2012, pp. 24–24.

[2] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "Simple-fying middlebox policy enforcement using SDN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, 2013, pp. 27–38.

[3] S. K. Fayazbakhsh, V. Sekar, M. Yu, and J. C. Mogul, "Flowtags: Enforcing network-wide policies in the presence of dynamic middlebox actions," in *Proc. 2nd ACM SIGCOMM Workshop HotSDN*, 2013, pp. 19–24.

[4] "Software-defined networking: The new norm for networks," *ONF White Paper*, 2012.

[5] A. Gember, et al., "Stratos a network-aware orchestration layer for virtual middleboxes in clouds," arXiv:1305.0209, 2013.

[6] H. Jamjoom, D. Williams, and U. Sharma, "Don't call them middleboxes, call them middlepipes," in *Proc. 3rd Workshop Hot Topics Software Defined Netw.*, 2014, pp. 19–24.

[7] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions in nfv," *2015 11th Int. Conf. Network Service Management (CNSM)*, pp. 50–56, 2015.

[8] P.-H. Huang, K.-W. Li, and C. H.-P. Wen, "NACHOS: Network-aware chains orchestration selection for NFV in SDN datacenter," in *Proc. 4th Int. Conf. Cloud Netw.*, 2015, pp. 205–208.

[9] T. Lukovszki and S. Schmid, "Online admission control and embedding of service chains," in *International Colloquium on Structural Information and Communication Complexity*. Berlin, Germany: Springer, 2014, pp. 104–118.

[10] Z. Cao, M. Kodialam, and T. Lakshman, "Traffic steering in software defined networks: Planning and online routing," in *Proc. ACM SIGCOMM Workshop Distrib. Cloud Comput.*s, 2014, pp. 65–70.

[11] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and T. Ming-Jer, "Deploying chains of virtual network functions on the relation between link and server usage," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.

[12] W. Hu, J. Gao, Y. Wang, O. Wu, and S. Maybank, "Online adaboost-based parameterized methods for dynamic distributed network intrusion detection," *IEEE Trans. Cybern.*, vol. 44, no. 1, pp. 66–82, Jan. 2014.

[13] N. A. Seresht and R. Azmi, "MAIS-IDS: A distributed intrusion detection system using multi-agent AIS approach," *Eng. Appl. Artif. Intell.*, vol. 35, pp. 286–298, 2014.

[14] S. Kent, "Ip authentication header," Tech. Rep., 2005.

[15] W. Ma, C. Medina, and D. Pan, "Traffic-aware placement of NFV middleboxes," in *Proc. IEEE Global Commun. Conf.*, 2015, pp. 1–6.

[16] S. Gu, Z. Li, C. Wu, and C. Huang, "An efficient auction mechanism for service chains in the NFV market," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.

[17] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of virtualized deep packet inspection functions in sdn," in *Proc. Military Commun. Conf.*, 2013, pp. 992–997.

[18] Y. Zhang, et al., "Steering: A software-defined networking for inline service chaining," in *Proc. IEEE Int. Conf. Netw. Protocols*, 2013, pp. 1–10.

[19] J. Liu, Y. Li, Y. Zhang, L. Su, and D. Jin, "Improve service chaining performance with optimized middlebox placement," *IEEE Trans. Serv. Comput.*, vol. 10, no. 4, pp. 560–573, Jul./Aug. 2015.

[20] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network function placement for nfv chaining in packet/optical datacenters," *J. Lightw. Technol.*, vol. 33, no. 8, pp. 1565–1570, 2015.

[21] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 1346–1354.

[22] Y. Li, L. Phan, and B. T. Loo, "Network functions virtualization with soft real-time guarantees," in *Proc. IEEE Conf. Comput. Commun.*, 2016, pp. 1–9.

[23] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *IEEE 4th Int. Conf. Cloud Netw.*, 2015, pp. 171–177.

[24] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *IEEE 3rd Int. Conf. Cloud Netw.*, 2014, pp. 7–13.

[25] S. Rajagopalan, D. Williams, H. Jamjoom, and A. Warfield, "Split/merge: System support for elastic execution in virtual middleboxes," in *Proc. 10th USENIX Conf. Netw. Syst. Des. Implementation*, vol. 13, 2013, pp. 227–240.

[26] L. Xie, Y. Shi, Y. T. Hou, W. Lou, H. D. Sherali, and S. F. Midkiff, "Multi-node wireless energy charging in sensor networks," *IEEE/ACM Trans. Netw.*, vol. 23, no. 2, pp. 437–450, Apr. 2015.

[27] X. Liu, et al.,"Top-k queries for categorized RFID systems," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2587–2600, Oct. 2017.

[28] Y. Gu, Y. Ji, J. Li, and B. Zhao, "ESWC: Efficient scheduling for the mobile sink in wireless sensor networks with delay constraint," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1310–1320, Jul. 2013.

[29] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," *IEEE Trans. Inform. Theory*, vol. 59, no. 10, pp. 6301–6327, Oct. 2013.

[30] Z. Shao, H. Zhang, M. Chen, and K. Ramchandran, "Reverse-engineering bittorrent: A markov approximation perspective," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2012, pp. 2996–3000.

[31] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint VM placement and routing for data center traffic engineering," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2012, pp. 2876–2880.

[32] H. Huang, S. Guo, W. Liang, K. Li, B. Ye, and W. Zhuang, "Near-optimal routing protection for in-band software-defined heterogeneous networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 11, pp. 2918–2934, 2016.

[33] H. Huang, S. Guo, J. Wu, and J. Li, "Joint middlebox selection and routing for software-defined networking," in *Proc. IEEE Int. Conf. Commun.*, 2016, pp. 1–6.

[34] H. Huang, S. Guo, J. Wu, and J. Li, "Service chaining for hybrid network function," *IEEE Trans. Cloud Comput.*, vol. PP, no. 99, p. 1, 2017, doi: 10.1109/TCC.2017.2721401.

[35] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

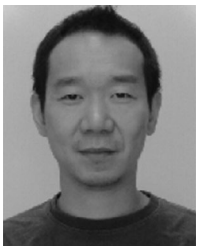
[36] F. P. Kelly, *Reversibility and Stochastic Networks*. Cambridge, U.K.: Cambridge Univ. Press, 2011.

[37] P. Diaconis and D. Stroock, "Geometric bounds for eigenvalues of markov chains," *Ann. Appl. Probability*, vol. 1, pp. 36–61, 1991.

[38] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov Chains and Mixing Times*. Providence, RI, USA: American Mathematical Soc., 2009.

Huawei Huang (M'16) received the PhD degree in computer science and engineering from the University of Aizu, Japan. His research interests mainly include optimization, algorithm design and analysis, mainly in the fields of software-defined networking (SDN), NFV, cloud computing, and wireless networks. He is a post-docotral research fellow of JSPS, and a member of the IEEE. Currently, he is a visiting scholar with the Hong Kong Polytechnic University.

Peng Li (S'10-M'12) received the BS degree from the Huazhong University of Science and Tech- nology, China, the MS and PhD degrees from the University of Aizu, Japan, in 2007, 2009, and 2012, respectively. He is currently an associate professor with the University of Aizu, Japan. His research interests mainly focus on wireless communication and networking, specifically wireless sensor networks, green and energy-efficient mobile networks, and cross-layer optimization for wireless networks. He also has interests include the cloud computing, big data processing and smart grid. He is a member of IEEE.



Song Guo (M'02-SM'11) received the PhD degree in computer science from the University of Ottawa. He is currently a full professor in the Department of Computing, The Hong Kong Polytechnic University. Prior to joining Polytechnic University, he was a full professor with the University of Aizu, Japan. His research interests include the mainly in the areas of cloud and green computing, big data, wireless networks, and cyber-physical systems. He has published more than 300 conference and journal papers in these areas and received multiple best paper awards from IEEE/ACM conferences. His research has been sponsored by JSPS, JST, MIC, NSF, NSFC, and industrial companies. He has served as an editor of several journals, including the *IEEE Transactions on Parallel and Distributed Systems*, the *Transactions on Emerging Topics in Computing*, the *IEEE Transactions on Green Communications and Networking*, the *IEEE Communications Magazine, and Wireless Networks*. He has been actively participating in international conferences serving as general chair and TPC chair. He is a senior member of the IEEE, a senior member of the ACM, and an IEEE Communications Society Distinguished Lecturer.

Weifa Liang (M'99-SM'01) received the BSc degree from the Wuhan University, China, the ME degree from the University of Science and Technology of China, and the PhD degree from the Australian National University, all in computer science, in 1984, 1989, and 1998. He is currently a professor in the Research School of Computer Science, the Australian National University. His research interests include design and analysis of energy-efficient routing protocols for wireless ad hoc and sensor networks, cloud computing, software-defined networking, design and analysis of parallel and distributed algorithms, approximation algorithms, combinatorial optimization, and graph theory. He is a senior member of the IEEE, a member of ACM.

Kun Wang (M'13-SM'17) received the B. Eng. and Ph.D. degree in the School of Computer, Nanjing University of Posts and Telecommunications, Nanjing, China, in 2004 and 2009, respectively. From 2013 to 2015, he was a Postdoc Fellow in Electrical Engineering Department, University of California, Los Angeles (UCLA), CA, USA. In 2016, he was a Research Fellow in the School of Computer Science and Engineering, the University of Aizu, Aizu-Wakamatsu City, Fukushima, Japan. He is currently a Research Fellow in the Department of Computing, the Hong Kong Polytechnic University, Hong Kong, China, and also a Full Professor in the School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing, China. He has published over 100 papers in referred international conferences and journals. He has received Best Paper Award at IEEE GLOBECOM16. He serves as Associate Editor of IEEE Access, Editor of Journal of Network and Computer Applications, Journal of Communications and Information Networks, EAI Transactions on Industrial Networks and Intelligent Systems and Guest Editors of IEEE Access, Future Generation Computer Systems, Peer-to-Peer Networking and Applications, and Journal of Internet Technology. He was the symposium chair/co-chair of IEEE IECON16, IEEE EEEIC16, IEEE WCSP16, IEEE CNCC17, etc. His current research interests are mainly in the area of big data, wireless communications and networking, smart grid, energy Internet, and information security technologies. He is a Senior Member of IEEE and Member of ACM.