Profit Maximization for Admitting Requests with Network Function Services in Distributed Clouds

Yu Ma, Weifa Liang, *Senior Member, IEEE*, Zichuan Xu, *Member, IEEE*, and Song Guo, *Senior Member, IEEE*

Abstract—Traditional networks employ expensive dedicated hardware devices as middleboxes to implement Service Function Chains of user requests by steering data traffic along middleboxes in the service function chains before reaching their destinations. Network Function Virtualization (NFV) is a promising virtualization technique that implements network functions as pieces of software in servers or data centers. The integration of NFV and Software Defined Networking (SDN) further simplifies service function chain provisioning, making its implementation simpler and cheaper. In this paper, we consider dynamic admissions of delay-aware requests with service function chain requirements in a distributed cloud with the objective to maximize the profit collected by the service provider, assuming that the distributed cloud is an SDN that consists of data centers located at different geographical locations and electricity prices at different data centers are different. We first formulate this novel optimization problem as a dynamic profit maximization problem. We then show that the offline version of the problem is NP-hard and formulate an integer linear programming solution to it. We third propose an online heuristic for the problem. We also devise an online algorithm with a provable competitive ratio for a special case of the problem where the end-to-end delay requirement of each request is negligible. We finally evaluate the performance of the proposed algorithms through experimental simulations. The simulation results demonstrate that the proposed algorithms are promising.

Index Terms—Network function virtualization, software defined networking, distributed data centers, online algorithms, service function chain consolidation, profit maximization, request admission scheduling

1 Introduction

Traditional networks are built upon proprietary software platforms tied onto proprietary hardware devices that evolved slowly [4]. Provisioning network services in traditional networks takes a great deal of time and effort, because different network hardware devices need to be acquired and configured in an appropriate manner. Also, user requests in traditional networks are implemented by dedicated hardware such as routers and associated (vendor-specified) protocols, which lack of flexibility, manageability, and efficiency [28]. Profits thus are being eroded by escalating operational expenses (OPEX) and capital expenses (CAPEX), and flat or declining revenue [21]. Network Function Virtualization (NFV) and Software Defined Networking (SDN) have been envisaged as next-generation networking paradigms to enable fast service deployment, inexpensive and flexible service provisioning in future communication networks [4]. Specifically, SDN introduces the concept of separation between the data plane and the control plane, thus provides more flexibility in steering network flow, while NFV enables highly optimized packet-processing network functions to run on commodity servers. Typically, each flow goes through *a sequence of network functions* in a specified order to perform necessary packet processing, such a sequence is termed as *the service function chain* (SFC) of the flow [23]. The integration of SDN and NFV enables flexible resource allocations to service function chains and optimizing resource utilization for user request admissions [28]. It thus can reduce network service providers' operational expenses and increase their profits significantly [5], [21], [29].

The adoption of SDN and NFV technologies poses several important challenges for network service providers. The first challenge is how to maximize their profits by admitting as many as requests while reducing their operational costs. This can be achieved by creating instances of service function chains in data centers and making use of the instances to implement NFVs of user requests. However, different user requests not only have different amounts of resource demands but also have different service function chains and end-to-end delay requirements. The second challenge is that, different data centers may have different energy consumption rates due to various specifications of servers. Also, the electricity prices of different geographical data centers may be quite different. The last challenge is how to dynamically admit user requests and find routing paths for the requests that include data centers to process their data traffic, assuming that the requests arrive one by one without the knowledge of future request arrivals. Tackling the aforementioned

- *Y. Ma and W. Liang are with the Research School of Computer Science, Australian National University, Canberra, ACT 2601, Australia. E-mail: yu.ma@anu.edu.au, wliang@cs.anu.edu.au.*
- *Z. Xu is with the School of Software, Dalian University of Technology, Dalian 116024, China. E-mail: z.xu@dlut.edu.cn.*
- *S. Guo is with the Hong Kong Polytechnic University Shenzhen Research Institute and Department of Computing, Hong Kong Polytechnic University, Hung Hom, Hong Kong. E-mail: song.guo@polyu.edu.hk.*

Manuscript received 3 Apr. 2018; revised 27 Aug. 2018; accepted 30 Sept. 2018. Date of publication 5 Oct. 2018; date of current version 10 Apr. 2019. (Corresponding author: Weifa Liang.)
Recommended for acceptance by R. Prodan.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2018.2874257

challenges requires novel frameworks, algorithms, and technologies, as conventional routing algorithms and protocols designed for traditional networks are not applicable to NFV-enabled SDNs.

The novelties of the work in this paper include: a novel optimization framework for dynamic admissions of NFV-enabled requests in an SDN that jointly considers computing and bandwidth resources provisioning at data centers and links to meet delay requirements of user requests, with the aim to maximize the profit of the network service provider; efficient online algorithms for the dynamic profit maximization problem are developed.

The main contributions of this paper are summarized as follows. We first formulate a dynamic profit maximization problem in a distributed cloud, by dynamically admitting delay-aware requests with service function chain requirements, assuming that the admission of each request is based on as pay-as-you-go, subject to various network resource capacities. We then show the offline version of the problem is NP-hard and formulate an integer linear programming (ILP) solution to it. We third propose an efficient heuristic for the dynamic profit maximization problem. We also devise an online algorithm with a provable competitive ratio for a special case of the problem where the end-to-end delay requirement is negligible. We finally conduct empirical evaluation on the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms are very promising.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 introduces notions, notations, and the problem definitions. Section 4 formulates an ILP solution to an offline version of the problem. Section 5 develops a heuristic for the problem, and Section 6 devises an online algorithm with a provable competitive ratio for a special case of the problem where the end-to-end delay requirement of each request can be negligent. Section 7 evaluates the performance of the proposed algorithms empirically, and Section 8 concludes the paper.

2 Related Work

There have been several studies on NFV-enabled user request routing in various networks [16], [17], [19], [30], [32], where a user request with a service function chain requirement is referred to as *the NFV-enabled request*. For example, Martins et al. [19] introduced a virtualization platform to improve network performance, by extending existing virtualization technologies to support the deployment of modular, virtual middleboxes on lightweight VMs. Wang et al. [30] investigated the problem of dynamic network function composition by proposing a distributed algorithm for the problem. Jia et al. [13] studied the dynamic provisioning of VNF service function chains across different data centers to minimize the operational cost of network service providers. They proposed an efficient online algorithm with a provable competitive ratio for the problem, using regularization techniques. Li et al. [16] investigated QoS-aware NFV-enabled user request routing problem by implementing the VNFs of the service function chain of each request in a single data center. Huang et al. [11] studied network throughput maximization problem by consolidating all VNFs in a service chain into a single server. They proposed two heuristic algorithms for the problem through striving for non-trivial tradeoffs between the accuracy of a solution and the running time to obtain the solution. Jia et al. [12] considered the operational cost minimization problem by utilizing pre-installed VNF instances. They also provided a prediction mechanism to dynamically create and remove VNF instances in data centers. Xu et al. [32] investigated the throughput maximization problem by making use of existing VNF instances, assuming that all requests are given in advance. Most aforementioned studies focused on developing either exact solutions for the problems by formulating Integer Linear Programming (ILP) solutions, or fast, scalable heuristic solutions.

There are several recent studies focusing on the profit maximization by admitting NFV-enabled requests in SDNs [7], [9], [24], [33], [34]. For example, Zhang et al. [34] devised a novel online stochastic auction mechanism for the network service chain provisioning and pricing for NFV service providers. They adopted the primal-dual optimization framework with a learning-based strategy for resource prices and developed an online auction mechanism with a provable competitive ratio to maximize the expected social welfare of the NFV ecosystem, including the NFV provider, customers and resource suppliers. Gu et al. [9] proposed an online auction mechanism for dynamic VNF service chain provisioning and pricing in a single data center, through incorporating the primal-dual approximation technique and Myersons characterization of truthful mechanism together. Racheg et al. [24] explored service function chain provisioning for distributed data centers with the aim to maximize the profit of network service providers. They proposed a heuristic for the profit maximization problem by enumerating all possible embedding paths for user requests that makes their algorithms unscalable, thereby restricting the applicability of the algorithm in the real world. Ghribi et al. [7] studied a profit maximization problem in SDNs by devising an algorithm that jointly considered VNF placement and traffic steering. Their algorithm deals with service function chains in multi-stages and deploys the VNF instances one by one, which leads to a sub-optimal solution. Yuan et al. [33] proposed a workload-aware revenue maximization approach by jointly considering VNF instance deployment and user request routing paths for each request admission. In addition, the problem of minimizing energy consumption of distributed cloud networks has been extensively explored in literature [2], [22], [35]. For example, Zhang et al. [35] observed that a large fraction of energy is wasted on maintaining excessive service capacity during low workload periods of data centers. They devised an online algorithm for minimizing the power consumption of data centers by dynamically switching on/off servers, according to resource demands of user requests. On the other hand, cloud service providers usually deploy their data centers in geographically different locations to meet ever-growing service demands of users from different regions. The electricity prices for data centers at different geographical locations are different, thus, the costs of energy consumption of these data centers are different too. Exploiting time-varying electricity prices in different regions has been shown a promising technique to minimize the operational costs of data centers, which has been explored in previous studies [24], [25], [26], [27], [31].

Unlike the aforementioned studies, in this paper we will study a dynamic profit maximization problem that admits NFV-enabled requests in SDNs through VNF instance placement while meeting delay requirements of user requests, assuming that requests arrive one by one without the knowledge of future request arrivals. We aim to maximize the profit of the network service provider through reducing its operational cost. This is achieved by incorporating varying energy consumption rates and electricity prices among data centers, and performing smart resource allocations to fully utilizing the resources in SDNs. This paper is an extended version of our conference paper [18].

3 Preliminaries

In this section, we first introduce the system model, notions and notations, and then define the problems precisely.

3.1 System Model

We consider a Software Defined Network (SDN) modelled by a directed graph $G = (S \cup V, E)$, where S is a set of SDN-enabled *switch nodes* and V is a set of *data centers* with rich computing resources that are geographically co-located with some of the switch nodes, and E is a set of *links* between SDN-enabled switches and between SDN-enabled switches and data centers. Denote by Q_v the set of active homogeneous servers in each data center $v \in V$, and for each server $s \in Q_v$, let C_s be its computing capacity. Thus, the computing capacity C_v of data center v is $C_v = \sum_{s \in Q_v} C_s$. Notice that different data centers may deploy different types of servers. Furthermore, the data traffic of each user request consumes link bandwidth and incurs data transmission delays on links in the routing path. Denote by B_e and d_e the bandwidth capacity and the delay on link $e \in E$.

3.2 Virtualized Network Functions and Service Function Chains

In conventional networks, network functions such as Firewalls, Intrusion Detection Systems (IDSes) and Load Balancers (LBs) are implemented by expensive, dedicated hardware middleboxes installed at network routers and switches [4]. Network function virtualization (NFV) that makes use of common x86 architecture to implement network functions as software components in data centers, has become a promising technology to significantly reduce the operational expenses (OPEX) and capital expenses (CAPEX) of network service providers, which introduces a new dimension for cost savings on dedicated hardware middleboxes and flexible deployment of network functions. The implementation of virtualized network functions (VNFs) in data centers consumes their computing resource. Also, different VNFs may have different processing delays depending on the resources that are allocated to them.

Since different requests may have different types of service function chains, establishing a routing path for each of the requests needs steering its flow to pass through the specified VNFs in its service function chain in a particular order. An ordered sequence of VNFs is defined as a *service function chain* (SFC). In this paper we assume that the VNFs of the SFC of each request is consolidated into a single VM in a data center for processing. This assumption has been adopted in [3], [10], [11].

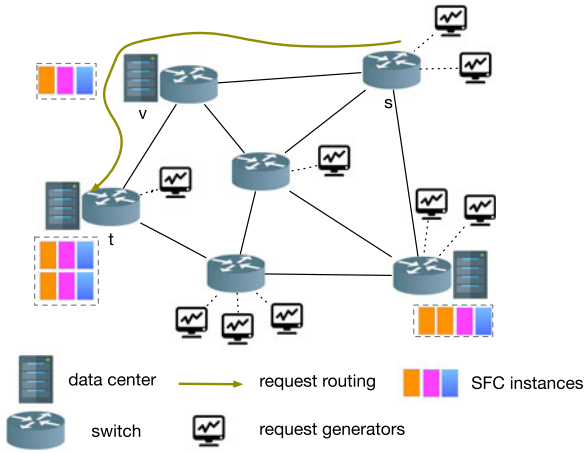


Fig. 1. An illustrative example of the admission of a user request in a Software Defined Network (SDN), where the request is issued at a source node s and its data traffic is processed at the data center v that contains the SFC instances of the request, and the processed data traffic is then routed to the destination t.

3.3 User Requests with Service Function Chain Requirements

Consider a *user request* $r_j = (s_j, t_j; SFC_j^{(k)}, b_j, \phi_j, D_j)$ that requires to transfer its data traffic from a source s_j to a destination t_j with a given packet rate $\phi_j > 0$, and the data traffic must pass through the instances of VNFs in its service function chain $SFC_j^{(k)}$. We assume that the service function chains among all requests can be classified into K types. We also assume that each instance of a VNF can process a basic packet rate [32], and a request r_j with ϕ_j packet rate implies it needs ϕ_j instances of its required type of service function chain and the amount b_j $(= \phi_j \cdot b^{(k)})$ of bandwidth resource, where $b^{(k)}$ is the bandwidth requirement of a basic data packet rate of a request with $SFC^{(k)}$. D_j is the end-to-end delay requirement of request r_j. We also assume that the data traffic of each request will be processed in a single data center. Denote by $SFC^{(k)}$ the service function chain of type k and $C(SFC^{(k)})$ the computing resource consumption of an instance of the service function chain of type k with $1 \leq k \leq K$. Assume that for each type of service function chain $SFC^{(k)}$, $C(SFC^{(k)}) > 1$. Each instance of service function chain $SFC_v^{(k)}$ in data center v has data processing rate $\mu_v^{(k)}$, which is also measured by the amount of basic packet rate. Fig. 1 is an illustrative example of a request admission in G.

3.4 End-to-End Delay Requirements of User Requests

Each request r_j has an end-to-end delay requirement D_j that specifies the maximum duration of per data packet from its source to its destination, which includes both the processing and queuing delay in a data center and the transmission delay along links.

Each data packet of an admitted request r_j will be queued in its service function chain instances in a data center before being processed, which will incur both the queuing and processing delays when each of the packets passes

through the service function chain instances. Each user request r_j with packet rate ϕ_j has ϕ_j instances of its required type of service function chain, we thus assume there is an $M/M/n$ queue for ϕ_j SFC instances of request r_j. The average queuing delay is $\frac{1}{\phi_j \cdot \mu_v^{(k)} - \phi_j} = \frac{1}{\phi_j \cdot (\mu_v^{(k)} - 1)}$.

Considering that the data processing rate of service function chain $SFC_v^{(k)}$ is $\mu_v^{(k)}$, the processing delay of $SFC_v^{(k)}$ in data center v is $\frac{1}{\mu_v^{(k)}}$. Denote by $d_j(SFC_v^{(k)})$ the processing and queuing delay of a packet at a data center v, i.e.,

$$d_j(SFC_v^{(k)}) = \frac{1}{\phi_j \cdot (\mu_v^{(k)} - 1)} + \frac{1}{\mu_v^{(k)}}. \tag{1}$$

Furthermore, traffic route for request r_j along a link $e \in E$ incurs a data transmission delay d_e. Denote by $P(s_j, v)$ the first segment of the routing path for the data traffic of request r_j from node s_j to data center $v \in V$. The data transmission delay along this segment $P(s_j, v)$ is $d_j(P(s_j, v)) = \sum_{e \in P(s_j, v)} d_e$. Similarly, the data transmission delay along the second segment $P(v, t_j)$ of the routing path is $d_j(P(v, t_j)) = \sum_{e \in P(v, t_j)} d_e$. The end-to-end delay of data transmission of request r_j along the routing path via data center v thus is

$$d(r_j) = d_j(P(s_j, v)) + d_j(SFC_v^{(k)}) + d_j(P(v, t_j)). \tag{2}$$

To meet the end-to-end delay requirement of r_j, we have $d(r_j) \leq D_j$.

3.5 Energy Consumption and Profit by Admitted Requests

Denote by $\mathcal{E}_v$ the energy consumption of data center $v \in V$, which is determined by the number of homogeneous servers $|Q_v|$ in v. We adopt a linear energy consumption model to capture the energy consumption of each server [24], [27], [35], that is, for a server $s \in Q_v$ in data center $v \in V$, the amount of energy $\mathcal{E}_s$ consumed by s per time unit is

$$\mathcal{E}_s = \mathcal{E}_v^{idle} + \eta_v \cdot \mathcal{U}_s, \tag{3}$$

where $\mathcal{U}_s = U_s / C_s$ is the utilization of computing resource in server s while U_s is the workload (the amount of computing resource being occupied at the moment) of server s and C_s is its computing capacity, $\mathcal{E}_v^{idle}$ is the amount of energy consumed per time unit when server s is *idle*, which usually is fixed. The energy consumption of a server s is proportional to its workload, and η_v is constant with $\eta_v > 0$, and the amount of energy $\mathcal{E}_v$ consumed in data center v per time unit thus is the sum of the amounts of energy consumed at all servers in Q_v, i.e.,

$$\begin{aligned} \mathcal{E}_v &= \sum_{s \in Q_v} \mathcal{E}_s = \sum_{s \in Q_v} (\mathcal{E}_v^{idle} + \eta_v \cdot \mathcal{U}_s) \\ &= |Q_v| \cdot \mathcal{E}_v^{idle} + \eta_v \sum_{s \in Q_v} \mathcal{U}_s, \\ &= |Q_v| \cdot \mathcal{E}_v^{idle} + \eta_v \cdot |Q_v| \cdot \frac{U_v}{C_v}, \end{aligned} \tag{4}$$

where $U_v = \sum_{s \in Q_v} U_s$ is the workload and C_v is the computing capacity of data center $v \in V$.

A network service provider typically charges its users by admitting their requests on a pay-as-you-go basis through adopting a common revenue collection model [24]. *The revenue* collected RV_j by admitting a request r_j is proportional to its computing and bandwidth resource demands, that is

$$RV_j = \lambda_1 \cdot \phi_j \cdot C(SFC^{(k)}) + \lambda_2 \cdot b_j, \tag{5}$$

where λ_1 and λ_2 are constant weights for computing and bandwidth prices, and ϕ_j is the packet rate of request r_j.

The profit $p(\mathcal{A})$ collected by admitting a set $\mathcal{A}$ of requests thus is the difference of the sum of the revenues collected to the cost of the amount of energy consumed at all data centers for the services [24], which is defined as follows:

$$p(\mathcal{A}) = \sum_{r_j \in \mathcal{A}} RV_j - \sum_{v \in V} \rho_v \cdot \mathcal{E}_v, \tag{6}$$

where ρ_v is the electricity price per unit energy at the location of data center $v \in V$, and RV_j and $\mathcal{E}_v$ are defined by Eqs. (5) and (4) respectively. Notice that the electricity prices at different locations and different time slots may be significantly different [31].

3.6 Problem Statements

Given an SDN $G = (S \cup V, E)$ and a set of NFV-enabled requests $\mathcal{R}$, *the profit maximization problem* is to admit as many requests in $\mathcal{R}$ as possible so that the profit collected by the network service provider is maximized, subject to network resource constraints on G.

We now define a dynamic version of the problem. Given an SDN $G = (S \cup V, E)$, let $r_1, r_2, \ldots r_j$ be a sequence of requests that arrive one by one without the knowledge of future arrivals, the *dynamic profit maximization problem* in G is to maximize the total profit collected by admitting as many as requests in the sequence while meeting the end-to-end delay requirement of each admitted request, subject to resource capacity constraints on both data centers and links.

The profit maximization problem is NP-hard, which can be shown by a reduction from the knapsack problem. The detailed reduction is omitted, due to space limitation.

3.7 Approximation and Competitive Ratios of Algorithms

A *γ-approximation algorithm* for a minimization problem P_1 is a polynomial time algorithm $\mathcal{A}$ that delivers an approximate solution for P_1 whose value is no more than γ times the optimal one for any instance of P_1 with $\gamma > 1$, where γ is termed as the approximation ratio of algorithm $\mathcal{A}$.

Let OPT and S be an optimal solution of the offline version of a problem and the solution delivered by an online algorithm $\mathcal{A}'$ for a maximization problem P_2 of dynamic request admissions, where requests arrive one by one without the knowledge of future arrivals. The *competitive ratio* of an online algorithm $\mathcal{A}'$ is ξ if $\frac{S}{OPT} \geq \xi$ for any instance I of problem P_2 with $0 < \xi < 1$.

4 ILP for the Profit Maximization Problem

In this section, we formulate an integer linear programming (ILP) solution to the profit maximization problem.

For each request $r_j \in \mathcal{R}$, a binary constant $a_j^k = 1$ if r_j is the type k request; $a_j^k = 0$ otherwise. A binary variable $x_j = 1$ if r_j is admitted; $x_j = 0$ otherwise. Recall that RV_j is the amount of revenue collected if request r_j is admitted. A binary variable $y_{j,v}$ is used to determine which data center will accommodate the instances of the service function chain of r_j, where $y_{j,v} = 1$ if it is implemented by the service function chain instances in data center v; $y_{j,v} = 0$ otherwise. For each request implementation in the network, we treat the data traffic of the request as a flow in G from its source to its destination. For clarity, denote by $\psi^-(u)$ and $\psi^+(u)$ the sets of incoming and outgoing edges of a node $u \in S \cup V$, respectively. The proposed ILP is described as follows:

$$maximize \sum_{r_j \in \mathcal{R}} RV_j \cdot x_j - \sum_{v \in V} \rho_v \cdot |Q_v| \cdot \mathcal{E}_v^{idle} - \sum_{v \in V} \rho_v \cdot |Q_v| \cdot \frac{\eta_v(\sum_{r_j \in \mathcal{R}} \sum_{k=1}^{K} a_j^k \cdot \phi_j C(SFC^{(k)}) \cdot y_{j,v})}{C_v}, \tag{7}$$

subject to:

$$\sum_{v \in V} y_{j,v} = x_j, \qquad \forall r_j \in \mathcal{R} \tag{8}$$

$$\sum_{e \in \psi^-(u)} z_j^{sv}(e) - \sum_{e \in \psi^+(u)} z_j^{sv}(e) = 0, \qquad \forall u \in S \setminus \{s_j\},\ r_j \in \mathcal{R} \tag{9}$$

$$\sum_{e \in \psi^+(u)} z_j^{vt}(e) - \sum_{e \in \psi^-(u)} z_j^{vt}(e) = 0, \qquad \forall u \in S \setminus \{t_j\},\ r_j \in \mathcal{R} \tag{10}$$

$$\sum_{e \in \psi^+(s_j)} z_j^{sv}(e) = x_j, \qquad \forall r_j \in \mathcal{R} \tag{11}$$

$$\sum_{e \in \psi^-(t_j)} z_j^{vt}(e) = x_j, \qquad \forall r_j \in \mathcal{R} \tag{12}$$

$$\sum_{e \in \psi^-(v)} z_j^{sv}(e) - \sum_{e \in \psi^+(v)} z_j^{sv}(e) = y_{j,v}, \forall v \in V, r_j \in \mathcal{R} \tag{13}$$

$$\sum_{e \in \psi^+(v)} z_j^{vt}(e) - \sum_{e \in \psi^-(v)} z_j^{vt}(e) = y_{j,v}, \forall v \in V, r_j \in \mathcal{R} \tag{14}$$

$$\sum_{e \in \psi^-(s_j)} z_j^{sv}(e) = 0, \qquad \forall r_j \in \mathcal{R} \tag{15}$$

$$\sum_{e \in \psi^+(t_j)} z_j^{vt}(e) = 0, \qquad \forall r_j \in \mathcal{R} \tag{16}$$

$$\sum_{e \in E} d_e(z_j^{sv}(e) + z_j^{vt}(e)) + \sum_{v \in V} d_j(SFC_v^{(k)}) \cdot y_{j,v} \le D_j, \qquad \forall r_j \in \mathcal{R} \tag{17}$$

$$\sum_{r_j \in \mathcal{R}} b_j \cdot (z_j^{sv}(e) + z_j^{vt}(e)) \le B_e,\ \forall e \in E \tag{18}$$

$$\sum_{r_j \in \mathcal{R}} \sum_{1 \le k \le K} a_j^k \cdot \phi_j C(SFC^{(k)}) \cdot y_{j,v} \le C_v,\ \forall v \in V \tag{19}$$

$$z_j^{sv}(e), z_j^{vt}(e) \in \{0, 1\}, \qquad \forall e \in E,\ r_j \in \mathcal{R} \tag{20}$$

$$x_j, y_{j,v} \in \{0, 1\}, \qquad \forall v \in V, r_j \in \mathcal{R} \tag{21}$$

Constraints (9) and (10) ensure the flow conservation at any switch node of a flow except the source and sink nodes of the flow. Constraints (11) and (12) enforce data traffic flow entering and leaving the network in accordance with the admission of r_j. As an edge $e \in E$ can be used in both the first segment $P(s_j, v)$ and the second segment $P(v, t_j)$ of the routing path of request r_j via data center v, two decision variables $z_j^{sv}(e)$ and $z_j^{vt}(e)$ are adopted to distinguish the usage of the edge in these two segments, respectively, where $z_j^{sv}(e) = 1$ if edge e is in $P(s_j, v)$; $z_j^{sv}(e) = 0$ otherwise. Similarly, $z_j^{vt}(e) = 1$ if edge e is in the second segment $P(v, t_j)$; $z_j^{vt}(e) = 0$ otherwise. Constraints (13) and (14) specify whether the service function chain of r_j is implemented in a data center $v \in V$. Constraints (15) and (16) ensure that no any fractional flow of r_j before it is processed by its SFC instance will enter its source node s_j and no any fractional flow of r_j processed by its SFC instance will leave from its destination t_j. Constraint (17) enforces the end-to-end delay requirement. Constraints (18) and (19) impose bandwidth and computing resource capacity constraints for all links and data centers in the network, respectively.

5 Online Algorithm for the Dynamic Profit Maximization Problem

In this section, we deal with the dynamic profit maximization problem. We first investigate *the delay-constrained shortest path problem via a specified node*, which will be used as a subroutine for the dynamic profit maximization problem. We then propose an efficient online algorithm for the problem.

5.1 A Delay-Constrained Shortest Path via a Specified Node

Given a graph $G = (S \cup V, E)$, a request r_j from a source s_j to a destination t_j with an end-to-end delay constraint D_j, and a specified node $v \in V$, the *delay-constrained shortest path via a specified node problem* in G is to find a minimum cost path (or route) from s_j to t_j that passes through the specified node v, while the end-to-end delay of the route is no greater than D_j. This problem is NP-hard as its special case the delay-constrained shortest path problem is NP-hard [14].

Assume that the implementation of the service function chain of a request $r_j = (s_j, t_j; SFC_j^{(k)}, b_j, \phi_j, D_j)$ by instances of network functions in a data center v, and the processing and queuing delay for the data traffic of r_j at v is $d_j(SFC_v^{(k)})$. To meet its end-to-end delay constraint D_j, its routing delay $d_j(P(s_j, v)) + d_j(P(v, t_j))$ from its source s_j to its destination t_j must be no greater than $D_j - d_j(SFC_v^{(k)})$. However, finding such a minimum cost routing path for request r_j from s_j to t_j via v is challenging, and existing algorithms for the delay-constrained shortest path problem is not applicable. We instead develop a novel algorithm for it. That is, for request r_j and data center $v \in V$, we construct an auxiliary graph $G'_{j,v}$ from G, we reduce the problem in G to the delay-constrained shortest path problem in $G'_{j,v} = (V', E')$, where $V' = \{u', u \mid \forall u \in S \cup V\}$, and $E' = \{\langle u', w' \rangle,\ \langle u, w \rangle \mid \forall \langle u, w \rangle \in E\}$. We then merge the specified data center node v and its copy v' into a single node v, i.e., $V' = V' \setminus \{v'\}$, $E' = E' \setminus (\bigcup_{\langle u', v' \rangle \in E'} \{\langle u', v' \rangle\} \cup \bigcup_{\langle v', u' \rangle \in E'} \{\langle v', u' \rangle\}) \cup (\bigcup_{\langle u', v' \rangle \in E'} \{\langle u', v \rangle\} \cup \bigcup_{\langle v', u' \rangle \in E'} \{\langle v, u' \rangle\})$ for any $u' \in V'$. Fig. 2 shows the construction of $G'_{j,v}$.

Having the constructed auxiliary graph $G'_{j,v}$, a delay-constrained shortest path $P'(s_j, t'_j)$ in $G'_{j,v}$ from node s_j to t'_j

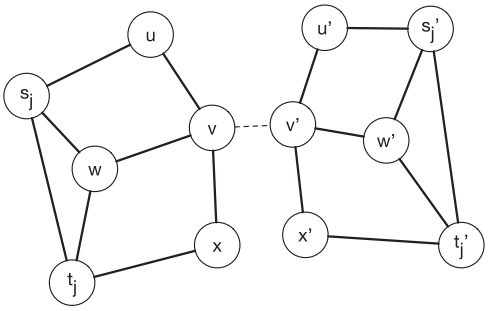


Fig. 2. An example of the construction of an auxiliary graph $G'_{j,v}$ for finding a delay-constrained shortest path via a specified node v for request r_j.

with a delay no greater than $D_j - d_j(SFC_v^{(k)})$ can then be found if it does exist, by applying the approximation algorithm due to Juttner et al. [14]. The pseudo-routing path (route) $P(s_j, t_j; v)$ in G for request r_j via data center v can then be derived from $P'(s_j, t'_j)$, where *a pseudo path (route)* is a path in which nodes and links can appear multiple times. The detailed description of the algorithm for finding a delay-constrained shortest path via a specified node is given in `Algorithm` 1.

Algorithm 1. Finding a Delay-Constrained Shortest Path in G via Node v for Request r_j

Input: An SDN $G = (S \cup V, E)$, a data center $v \in V$, a request $r_j = (s_j, t_j; SFC_j^{(k)}, b_j, \phi_j, D_j)$.

Output: A delay-constrained shortest path via data center v; if it does not exist, data center v cannot be used for implementing service function chain for r_j without violating its delay constraint.

1: Construct the auxiliary graph $G'_{j,v}$ from G;
2: Find an approximate delay-constrained shortest path $P'(s_j, t'_j)$ in $G'_{j,v}$ from s_j to t'_j subject to the delay constraint on its routing path such that $d_j(P(s_j, v)) + d_j(P(v, t_j)) \leq D_j - d_j(SFC_v^{(k)})$, by applying the algorithm due to Juttner et al. [14];
3: **if** $P'(s_j, t'_j)$ in $G'_{j,v}$ exists **then**
4: A routing path $P(s_j, t_j; v)$ in G is derived from $P'(s_j, t'_j)$, by replacing each edge in $P'(s_j, t'_j)$ with its corresponding edge in G;
5: **else**
6: Request r_j cannot make use of data center v for service function chain implementation.

Lemma 1. *(i) If there is a directed path in $G'_{j,v}$ from s_j to t'_j, it must pass through node v, i.e., v is an articulation point in $G'_{j,v}$; (ii) any delay-constrained shortest path $P'(s_j, t'_j)$ in $G'_{j,v}$ from s_j to t'_j cannot pass through node v twice; and (iii) each edge or node in G appears at most twice in the route $P(s_j, t_j; v)$ of G which is derived from path $P'(s_j, t'_j)$.*

Proof. We first show claim (i) by contradiction. Assume that there is a directed path from s_j to t'_j that does not pass through node v. We remove v and its incident edges from $G'_{j,v}$, this will not impact the directed path since v is not on it. However, the removal of v from $G'_{j,v}$ will lead to the resulting graph disconnected and nodes s_j and $t_{j'}$ will not be in the same connected component. Then, there is not any directed path from s_j to t'_j, which forms a contradiction.

We then prove claim (ii). Assume that there is such a delay-constrained shortest path $P'(s_j, t'_j)$ from s_j to t'_j that passes through v twice. Then, $P'(s_j, t'_j)$ must contain a directed cycle with v in it. If we break the cycle by removing some of its edges and nodes except v while keeping t'_j reachable from s_j. The length of the resulting path from s_j to t'_j thus is shorter than that of $P'(s_j, t'_j)$, which contradicts the fact that $P'(s_j, t'_j)$ is a shortest delay-constrained path from s_j to t'_j.

We finally show claim (iii). It can be seen that $P'(s_j, t'_j)$ is a simple path in $G'_{j,v}$. Each edge $\langle u, w\rangle \in E$ is contained in the route $P(s_j, t_j; v)$ in G derived from $P'(s_j, t'_j)$ if $\langle u, w\rangle \in P'(s_j, t'_j)$ or $\langle u', w'\rangle \in P'(s_j, t'_j)$. Thus, each edge or node in G appears at most twice in $P(s_j, t_j; v)$. □

Theorem 1. *Given a graph $G = (S \cup V, E)$, a request r_j, and a specified node $v \in V$, there is an approximation algorithm with an approximation ratio of $(1 + \epsilon)$ for the delay-constrained shortest path problem via a specified node, which takes $O(|E|^2 \cdot \log^4 |E|)$ time, where ϵ is a constant with $0 < \epsilon \leq 1$.*

Proof. For a request r_j and any specified node $v \in V$, the construction of auxiliary graph $G'_{j,v}$ from G takes $O(|S \cup V| + |E|)$ time. Finding an approximate delay-constrained shortest path in $G'_{j,v}$ takes time $O(|E'_{j,v}|^2 \log^4 |E'_{j,v}|)$, by applying the $(1 + \epsilon)$-approximation algorithm due to Juttner et al. [14], where $|E'_{j,v}| (= 2|E|)$ is the number of edges in $G'_{j,v}$. Thus, the running time of `Algorithm` 1 is $O(|E'_{j,v}|^2 \cdot \log^4 |E'_{j,v}| + O(|S \cup V| + |E|)) = O(|E|^2 \cdot \log^4 |E|)$. □

5.2 Online Algorithm

We now propose an online algorithm for the dynamic profit maximization problem.

We start by introducing a usage weight model to measure various network resource consumptions. Specifically, given the dynamics of user requests and networks, there is a need of a cost model to capture dynamic consumption of various resources in the network in order to better guide the admissions of future requests and utilize the resources. Intuitively, overloaded resources usually have higher probabilities of violating the resource demands of admitted requests due to the high dynamics of resource consumptions. This will affect the admissions of future requests eventually. Thus underloaded resources should be encouraged to be used while overloaded resources should be restricted (discouraged) to be used when conducting resource allocations to meet resource demands of admitting requests. If a specific resource has been highly utilized, it should be less used in the future by assigning it a higher usage weight; otherwise (a specific resource has rarely been used), it should be encouraged to be used by assigning it a lower usage weight. The usage weight model is given as follows.

When request r_j arrives, for each data center node $v \in V$, the usage weight $c_v(j)$ of using the computing resource at data center v by request r_j is defined as

$$c_v(j) = \rho_v \cdot C_v(\alpha^{1 - \frac{C_v(j)}{C_v}} - 1) \qquad \forall v \in V, \tag{22}$$

where α is a tuning parameter with $\alpha > 1$, $C_v(j)$ is the residual computing resource at data center v when request r_j arrives with $C_v(j) = C_v(j-1) - \phi_j \cdot C(SFC^{(k)})$ if request r_j admitted and $C_v(0) = C_v$, ϕ_j is the packet rate of request r_j, and ρ_v is the electricity price per unit energy at the location of data center v.

For each link $e \in E$, the usage weight $c_e(j)$ of using the bandwidth resource at link e in the route for request r_j can be defined similarly, that is

$$c_e(j) = B_e(\beta^{1-\frac{B_e(j)}{B_e}} - 1) \qquad \forall e \in E, \tag{23}$$

where β is a tuning parameter with $\beta > 1$, and $B_e(j)$ is the residual bandwidth at link e when request r_j arrives with $B_e(j) = B_e(j-1) - b_j$ if r_j admitted and $B_e(0) = B_e$.

Denote by ρ_{max} and ρ_{min} the maximum and minimum electricity prices among the locations of all data centers, i.e., $\rho_{max} = \max_{v \in V}\{\rho_v\}$ and $\rho_{min} = \min_{v \in V}\{\rho_v\}$.

When request r_j arrives, *the normalized usage weights* $\omega_v(j)$ at each data center $v \in V$ and $\omega_e(j)$ at each link $e \in E$ of G for request r_j are defined as follows:

$$\omega_v(j) = \frac{c_v(j)}{\rho_{max} \cdot C_v} = \frac{\rho_v}{\rho_{max}} \cdot (\alpha^{1-\frac{C_v(j)}{C_v}} - 1) \quad \forall v \in V, \tag{24}$$

$$\omega_e(j) = \frac{c_e(j)}{B_e} = \beta^{1-\frac{B_e(j)}{B_e}} - 1 \qquad \forall e \in E. \tag{25}$$

For the admission of a request r_j, denote by $G_j = (S \cup V, E; \omega_v(j), \omega_e(j))$ a weighted version of a subgraph of G by removing those nodes and links that do not have enough residual resources to accommodate r_j, in which its data center nodes and edges are assigned with the defined normalized usage weights. Then, for each data center $v \in V$, a delay-constrained shortest path (route) $P(s_j, t_j; v)$ in G_j from s_j to t_j via v is found by applying `Algorithm` 1. One delay-constrained shortest path in G via a data center v' is identified if $P(s_j, t_j; v') = \min_{v \in V}\{P(s_j, t_j; v)\}$.

To avoid admitting request r_j that demands too much resources (leading to a higher implementation usage weight, thereby undermining the performance of the SDN), we then adopt the following *admission control policy*: request r_j will be rejected if either (i) $\omega_{v'}(j) > \sigma_1$ for the chosen data center $v' \in V$ or (ii) $\sum_{e \in P(s_j, t_j; v')} \omega_e(j) > \sigma_2$, where σ_1 and σ_2 are admission control thresholds of computing and bandwidth resources, respectively, where $\sigma_1 = (n-1) \cdot \rho_{min}/\rho_{max}$, $\sigma_2 = n - 1$, and $n = |S \cup V|$.

The detailed algorithm for the dynamic profit maximization problem is given in `Algorithm` 2.

The time complexity of the proposed online algorithm, `Algorithm` 2, is stated in the following theorem.

Theorem 2. *Given an SDN $G = (S \cup V, E)$ with a set V of data centers, there is an online algorithm, `Algorithm` 2, for the dynamic profit maximization problem, which delivers a feasible solution for each request in $O(|E|^2 \cdot \log^4|E| \cdot |V|)$ time.*

Proof. We analyze the time complexity of `Algorithm` 2. The assignment of weights on nodes and edges of G_j takes $O(|S \cup V| + |E|)$ time. For each request r_j and each data center v, finding a delay-constrained shortest path via a node v takes $O(|E|^2 \log^4|E|)$ time. Finding one delay-constrained shortest path which is the minimum one among all such delay-constrained shortest paths thus takes $O(|E|^2 \cdot \log^4|E| \cdot |V|)$ time. `Algorithm` 2 therefore takes time $O(|E|^2 \cdot \log^4|E| \cdot |V|)$ to determine whether a request should be admitted. □

Algorithm 2. Maximizing the Profit for the Dynamic Profit Maximization Problem

Input: An SDN $G = (S \cup V, E)$ with a set V of data centers, a sequence of requests $r_j = (s_j, t_j; SFC_j^{(k)}, b_j, \phi_j, D_j) \in \mathcal{R}$, assuming that requests arrive one by one without the knowledge of future arrivals.

Output: A solution to maximize the profit. If a request r_j is admitted, a data center v' to implement its service function chain and its routing path (route) from s_j to t_j via v' will be delivered.

1: $\mathcal{A} \leftarrow \emptyset$;
2: **while** request r_j arrives **do**
3: Construct the auxiliary graph G_j for request r_j;
4: Find a delay-constrained shortest path (route) $P(s_j, t_j; v')$ via data center v' among all delay-constrained shortest paths via all different data centers in V, by invoking Algorithm 1 for r_j via each data center $v \in V$ such that the one via node v' is the minimum one; if no such path exists, reject r_j;
5: Determine whether r_j to be accepted or rejected by the defined admission control policy;
6: **if** r_j is accepted **then**
7: $\mathcal{A} \leftarrow \mathcal{A} \cup \{r_j\}$;
8: **return** $\mathcal{A}$.

In practice, almost all SDNs are planar, and the degree of each node in such a network is a small constant, i.e., $|E| = O(|V|)$. The time complexity of the algorithm thus can be further reduced to $O(|V|^3 \cdot \log^4|V|)$.

6 Online Algorithm for a Special Dynamic Profit Maximization Problem

In this section we consider a special case of the dynamic profit maximization problem where the end-to-end delay constraint can be neglected, for which we devise an online algorithm with a provable competitive ratio.

6.1 Online Algorithm with a Provable Competitive Ratio

The algorithm is to perform a minor modification to Step 4 in `Algorithm` 2. That is, a shortest path $P(s_j, t_j; v')$ instead of a delay-constrained shortest path, in G_j will be found. The detailed description of `Algorithm` 3 through the modification to `Algorithm` 2 is given as follows.

6.2 The Competitive Ratio and Time Complexity Analysis of the Online Algorithm

The rest is to analyze the performance of `Algorithm` 3. We first show the upper bound on the total usage weight of admitted requests when request r_j arrives in terms of the defined usage weight model by Eqs. (22) and (23). We then provide a lower bound on the usage weight of a rejected request by `Algorithm` 3 but admitted by an optimal offline

algorithm. We finally derive the competitive ratio of Algorithm 3.

Algorithm 3. Maximizing the Profit for a Special Dynamic Profit Maximization Problem without End-to-End Delay Constraints

Input: An SDN $G = (S \cup V, E)$ with a set V of data centers, a sequence of requests $r_j = (s_j, t_j; SFC_j^{(k)}, b_j, \phi_j) \in \mathcal{R}$, assuming that requests arrive one by one without the knowledge of future arrivals.

Output: A solution to maximize the profit. If a request r_j is admitted, a data center v' to implement its service function chain and its routing path (route) from s_j to t_j via v' will be delivered.

1: $\mathcal{A} \leftarrow \emptyset$;
2: **while** request r_j arrives **do**
3: Construct the auxiliary graph G_j for request r_j;
4: Find a shortest path (route) $P(s_j, t_j; v')$ via data center v' which is the minimum one among all shortest paths via all different data centers in V, by invoking the modified version of Algorithm 1 (i.e., set delay requirement D_j of r_j as $+\infty$) for r_j via each data center $v \in V$ such that the path length via node v' is the minimum one; if no such path exists for all data centers, reject r_j;
5: Determine whether r_j to be accepted or rejected by the defined admission control policy;
6: **if** r_j is accepted **then**
7: $\mathcal{A} \leftarrow \mathcal{A} \cup \{r_j\}$;
8: **return** $\mathcal{A}$.

We start by showing the upper bound on the total usage weight of all admitted requests on the arrival of request r_j by Algorithm 3 in the following lemma.

Lemma 2. *Given an SDN $G = (S \cup V, E)$ with computing capacity C_v at each data center $v \in V$ and link bandwidth capacity B_e at each link $e \in E$, denote by $\mathcal{A}(j)$ the set of requests admitted by Algorithm 3 prior to the arrival of request r_j. The sum of the usage weights of all data centers and links in G defined in Eqs. (22) and (23) prior to the arrival of r_j thus are*

$$\sum_{v \in V} c_v(j) \leq \rho_{max} \cdot \mathbb{C}(j)(\sigma_1 + n - 1)\log\alpha, \tag{26}$$

and

$$\sum_{e \in E} c_e(j) \leq \mathbb{B}(j)(\sigma_2 + n - 1)\log\beta, \tag{27}$$

respectively, provided that $\phi_{j'} \cdot C(SFC^{(k')}) \leq \frac{\min_{v \in V}\{C_v\}}{\log\alpha}$ and $b_{j'} \leq \frac{\min_{e \in E}\{B_e\}}{\log\beta}$ with $1 \leq j' \leq j$, where $\mathbb{C}(j)$ $(= \sum_{r_{j'} \in \mathcal{A}(j)} \phi_{j'} \cdot C(SFC^{(k')}))$ and $\mathbb{B}(j)$ $(= \sum_{r_{j'} \in \mathcal{A}(j)} b_{j'})$ are the accumulated amounts of computing and bandwidth resources being occupied by the admitted requests in $\mathcal{A}(j)$ at the moment, respectively.

Proof. Consider a request $r_{j'} \in \mathcal{A}(j)$ admitted by the online algorithm Algorithm 3 and its service function chain is implemented in data center $v \in V$, the usage weight of data center v in G by admitting request $r_{j'}$ is

$$\begin{aligned}
&c_v(j'+1) - c_v(j') \\
&= \rho_v \cdot C_v(\alpha^{1-\frac{C_v(j'+1)}{C_v}} - 1) - \rho_v \cdot C_v(\alpha^{1-\frac{C_v(j')}{C_v}} - 1) \\
&= \rho_v \cdot C_v(\alpha^{1-\frac{C_v(j'+1)}{C_v}} - \alpha^{1-\frac{C_v(j')}{C_v}}) \\
&= \rho_v \cdot C_v\alpha^{1-\frac{C_v(j')}{C_v}}(\alpha^{\frac{C_v(j')-C_v(j'+1)}{C_v}} - 1) \\
&= \rho_v \cdot C_v\alpha^{1-\frac{C_v(j')}{C_v}}(\alpha^{\frac{\phi_{j'} \cdot C(SFC^{(k')})}{C_v}} - 1) \\
&= \rho_v \cdot C_v\alpha^{1-\frac{C_v(j')}{C_v}}(2^{\frac{\phi_{j'} \cdot C(SFC^{(k')})}{C_v}\log\alpha} - 1) \\
&\leq \rho_v \cdot \alpha^{1-\frac{C_v(j')}{C_v}} \cdot \phi_{j'} \cdot C(SFC^{(k')}) \cdot \log\alpha.
\end{aligned} \tag{28}$$

Notice that Inequality (28) holds due to that $2^x - 1 \leq x$ for $0 \leq x \leq 1$.

If an edge $e \in E$ is in the routing path $P(s_{j'}, t_{j'}; v)$ of $r_{j'}$ for request $r_{j'}$ admission by Algorithm 3, the usage weight of edge e in the admission of request $r_{j'}$ is

$$\begin{aligned}
&c_e(j'+1) - c_e(j') \\
&= B_e(\beta^{1-\frac{B_e(j'+1)}{B_e}} - 1) - B_e(\beta^{1-\frac{B_e(j')}{B_e}} - 1) \\
&= B_e(\beta^{1-\frac{B_e(j'+1)}{B_e}} - \beta^{1-\frac{B_e(j')}{B_e}}) \\
&= B_e\beta^{1-\frac{B_e(j')}{B_e}}(\beta^{\frac{B_e(j')-B_e(j'+1)}{B_e}} - 1) \\
&= B_e\beta^{1-\frac{B_e(j')}{B_e}}(\beta^{\frac{b_{j'}}{B_e}} - 1), \\
&= B_e\beta^{1-\frac{B_e(j')}{B_e}}(2^{\frac{b_{j'}}{B_e}\log\beta} - 1) \\
&\leq \beta^{1-\frac{B_e(j')}{B_e}} \cdot b_{j'} \cdot \log\beta.
\end{aligned} \tag{29}$$

We then calculate the sum of the usage weights of all data centers and all edges in G when admitting request $r_{j'}$. If a data center was not chosen for implementing the service function chain of $r_{j'}$, its usage weight does not change after the admission of $r_{j'}$; similarly, if an edge is not included in $P(s_{j'}, t_{j'}; v)$, its usage weight remains unchanged after the admission of $r_{j'}$. The difference of the sum of the usage weights of data centers before and after admitting $r_{j'}$ is

$$\begin{aligned}
&\sum_{v \in V} c_v(j'+1) - c_v(j') \\
&= c_v(j'+1) - c_v(j') \\
&\leq \rho_v \cdot \alpha^{1-\frac{C_v(j')}{C_v}} \cdot \phi_{j'} \cdot C(SFC^{(k')}) \cdot \log\alpha, \quad \text{by (28)} \\
&= \rho_v \cdot \phi_{j'} \cdot C(SFC^{(k')})\log\alpha((\alpha^{1-\frac{C_v(j')}{C_v}} - 1) + 1) \\
&= \rho_v \cdot \phi_{j'} \cdot C(SFC^{(k')})\log\alpha(\omega_v(j') \cdot \frac{\rho_{max}}{\rho_v} + 1) \\
&= \phi_{j'} \cdot C(SFC^{(k')})\log\alpha(\omega_v(j') \cdot \rho_{max} + \rho_v) \\
&\leq \rho_{max} \cdot \phi_{j'} \cdot C(SFC^{(k')}) \cdot (\sigma_1 + 1)\log\alpha \\
&\leq \rho_{max} \cdot \phi_{j'} \cdot C(SFC^{(k')}) \cdot (\sigma_1 + n - 1)\log\alpha.
\end{aligned} \tag{30, 31}$$

Inequality (30) holds due to the fact that if request $r_{j'}$ is admitted, then Condition (i) of the admission control policy is satisfied, i.e., $\omega_v(j') \leq \sigma_1$.

The difference of the sum of the usage weights of all edges in G before and after admitting $r_{j'}$ is

$$\begin{aligned}
&\sum_{e\in E}(c_e(j'+1)-c_e(j'))\\
&=\sum_{e\in P(s_{j'},t_{j'};v)}(c_e(j'+1)-c_e(j'))\\
&\le\sum_{e\in P(s_{j'},t_{j'};v)}(\beta^{1-\frac{B_e(j')}{B_e}}\cdot b_{j'}\cdot\log\beta),\quad\text{by (29)}\\
&=b_{j'}\cdot\log\beta\sum_{e\in P(s_{j'},t_{j'};v)}((\beta^{1-\frac{B_e(j')}{B_e}}-1)+1)\\
&=b_{j'}\cdot\log\beta\left(\sum_{e\in P(s_{j'},t_{j'};v)}(\beta^{1-\frac{B_e(j')}{B_e}}-1)+\sum_{e\in P(s_{j'},t_{j'};v)}1\right)\\
&=b_{j'}\cdot\log\beta\left(\sum_{e\in P(s_{j'},t_{j'};v)}\omega_e(j')+\sum_{e\in P(s_{j'},t_{j'};v)}1\right)\\
&\le b_{j'}\cdot(\sigma_2+n-1)\log\beta.
\end{aligned}\tag{32}$$

The sum of the usage weights of all data centers prior to the arrival of request r_j thus is

$$\begin{aligned}
&\sum_{v\in V}c_v(j)=\sum_{j'=1}^{j-1}\sum_{v\in V}c_v(j'+1)-c_v(j')\\
&=\sum_{r_{j'}\in\mathcal{A}(j)}\sum_{v\in V}c_v(j'+1)-c_v(j')\\
&\le\sum_{r_{j'}\in\mathcal{A}(j)}(\rho_{max}\cdot\phi_{j'}\cdot C(SFC^{(k')})(\sigma_1+n-1)\log\alpha),\quad\text{by (31)}\\
&=\rho_{max}\cdot(\sigma_1+n-1)\cdot\log\alpha\sum_{r_{j'}\in\mathcal{A}(j)}\phi_{j'}\cdot C(SFC^{(k')})\\
&=\rho_{max}\cdot\mathbb{C}(j)\cdot(\sigma_1+n-1)\log\alpha,
\end{aligned}\tag{33}$$

and the sum of the usage weights of all edges in G prior to the arrival of request r_j is

$$\begin{aligned}
&\sum_{e\in E}c_e(j)=\sum_{j'=1}^{j-1}\sum_{e\in E}c_e(j'+1)-c_e(j')\\
&=\sum_{r_{j'}\in\mathcal{S}(j)}\sum_{e\in E}c_e(j'+1)-c_e(j')\\
&\le\sum_{r_{j'}\in\mathcal{S}(j)}(b_{j'}\cdot(\sigma_2+n-1)\log\beta),\quad\text{by (32)}\\
&=(\sigma_2+n-1)\cdot\log\beta\sum_{r_{j'}\in\mathcal{A}(j)}b_{j'}\\
&=\mathbb{B}(j)\cdot(\sigma_2+n-1)\log\beta.
\end{aligned}\tag{34}$$

□

We now provide a lower bound on the usage weight of a rejected request by `Algorithm 3` but admitted by *an optimal offline algorithm OPT*. Before we proceed, we choose appropriate values for α and β in Eqs. (22) and (23) prior to the arrival of any request r_j with its $SFC^{(k)}$ as follows:

$$2n\le\alpha\le\min_{v\in V}\{2^{\frac{C_v}{\phi_j\cdot C(SFC^{(k)})}}\},\tag{35}$$

$$2n\le\beta\le\min_{e\in E}\{2^{\frac{B_e}{b_j}}\}.\tag{36}$$

By Inequality (35), for any $v'\in V$, any request $r_j\in\mathcal{R}$, any edge $e'\in E$, and any k' with $1\le k'\le K$, we have

$$2^{\frac{C_{v'}}{\phi_{j'}\cdot C(SFC^{(k')})}}\ge\min_{v\in V}\{2^{\frac{C_v}{\phi_j\cdot C(SFC^{(k)})}}\}\ge\alpha,\tag{37}$$

$$2^{\frac{B_{e'}}{b_{j'}}}\ge\min_{e\in E}\{2^{\frac{B_e}{b_j}}\}\ge\beta.\tag{38}$$

Lemma 3. *Let $\mathcal{T}(j)$ be the set of requests rejected by `Algorithm 3` but admitted by algorithm OPT prior to the arrival of request r_j. Let $P_{OPT}(s_{j'},t_{j'};v^*)$ be the routing path in $G_{j'}$ by algorithm OPT for any request $r_{j'}\in\mathcal{T}(j)$ with $1\le j'\le j$, and assuming that the data center v^* is identified for the implementation of the service function chain $SFC^{(k)}$ of request $r_{j'}$. Then*

$$\sum_{e\in P_{OPT}(s_{j'},t_{j'};v^*)}\omega_e(j')+\omega_v(j')\ge\min\{\sigma_1,\sigma_2\}=\sigma_1.\tag{39}$$

Proof. Consider a request r'_j, which is rejected by `Algorithm 3` by one of the three cases: (i) there is no sufficient computing resource in any data center for implementing the service function chain of the request; (ii) there is no sufficient bandwidth in G for routing its data traffic; and (iii) the normalized usage weight (defined by Eq. (24)) of the selected data center to implement its service function chain is too high, or the sum of the normalized usage weights of edges (defined by Eq. (25)) in its routing path is too high.

Case (i). Let v^* be the data center selected to implement the service function chain of request r'_j of type k'. As the request is rejected by `Algorithm 3` due to insufficient available computing resource, this implies that $C_{v^*}(j')<\phi_{j'}\cdot C(SFC^{(k')})$. We thus have

$$\begin{aligned}
\omega_v(j')&=\omega_{v^*}(j')\\
&=\frac{\rho_v}{\rho_{max}}\cdot(\alpha^{1-\frac{C_{v^*}(j')}{C_{v^*}}}-1)\\
&>\frac{\rho_{min}}{\rho_{max}}\cdot(\alpha^{1-\frac{\phi_{j'}\cdot C(SFC^{(k')})}{C_{v^*}}}-1)\\
&\ge\frac{\rho_{min}}{\rho_{max}}\cdot(\alpha^{1-\frac{1}{\log\alpha}}-1),\quad\text{by (37)}\\
&=\frac{\rho_{min}}{\rho_{max}}\cdot(\frac{\alpha}{2}-1)\ge\sigma_1.
\end{aligned}$$

Case (ii). If request $r_{j'}$ is rejected, then there is an edge $e'\in P(s_{j'},t_{j'};v^*)$ that does not have sufficient residual bandwidth to accommodate the request. This implies that $B_{e'}(j')<b_{j'}$. Therefore, the length of $P(s_{j'},t_{j'};v^*)$ is greater than σ_2

$$\begin{aligned}
\textstyle\sum_{e\in P(s_{j'},t_{j'};v^*)}w_e(j')&\ge w_{e'}(j')\\
&=\beta^{1-\frac{B_{e'}(j')}{B_{e'}}}-1\\
&>\beta^{1-\frac{b_{j'}}{B_{e'}}}-1,\quad\text{since } B_{e'}(j')<b_{j'}\\
&\ge\beta^{1-\frac{1}{\log\beta}}-1,\quad\text{by (38)}\\
&=\frac{\beta}{2}-1\ge\sigma_2.
\end{aligned}$$

Case (iii). Although Algorithm 3 is able to deliver a shortest path $P(s_{j'}, t_{j'}; v)$ for request $r_{j'}$, it rejects $r_{j'}$ due to the fact that the implementation usage weight of $r_{j'}$ is too expensive, violating condition (i) or (ii) of the admission control policy, i.e., either $\omega_v(j') > \sigma_1$ or $\sum_{e \in P(s_{j'}, t_{j'}; v)} \omega_e(j') > \sigma_2$. Consequently, $\sum_{e \in P_{OPT}(s_{j'}, t_{j'}; v^*)} \omega_e(j') + \omega_v(j') \geq \min\{\sigma_1, \sigma_2\}$, Lemma 3 thus follows. □

We finally analyze the competitive ratio of Algorithm 3 in the following theorem.

Theorem 3. *Given an SDN $G = (S \cup V, E)$ with computing capacity C_v of each data center $v \in V$ and bandwidth capacity B_e for each link $e \in E$, assume that there is a sequence of requests with each request $r_j = (s_j, t_j; SFC_j^{(k)}, b_j, \phi_j)$ arriving one by one without the knowledge of future arrivals, there is an online algorithm, Algorithm 3, with a competitive ratio of $O(\log n)$ for the special case of the dynamic profit maximization problem without the end-to-end delay constraint, and the algorithm takes $O(|S \cup V|^2 \cdot |V|)$ time to admit each request when $\alpha = 2n$ and $\beta = 2n$, where $n = |S \cup V|$.*

Proof. Let $\mathbb{C}_{OPT}(j)$ and $\mathbb{B}_{OPT}(j)$ be the total amounts of computing and bandwidth resources being occupied by the admitted requests by the optimal offline algorithm OPT prior to the arrival of request r_j, and let $P_{OPT}(s_{j'}, t_{j'}; v^*)$ be the optimal routing path for request $r_{j'} \in \mathcal{T}(j)$ and $v^* \in V$ the data center to implement the service function chain of $r_{j'}$. Recall that $\mathbb{B}(j) = \sum_{r_{j'} \in \mathcal{A}(j)} b_{j'}$ and $\mathbb{C}(j) = \sum_{r_{j'} \in \mathcal{A}(j)} \phi_{j'} \cdot C(SFC^{(k')})$ are the accumulated bandwidth and computing resources being occupied by admitted requests in $\mathcal{A}(j)$ at the moment, respectively. Then

$$
\begin{aligned}
&(n-1)(\mathbb{B}_{OPT}(j) - \mathbb{B}(j)) \leq (n-1) \sum_{r_{j'} \in \mathcal{T}(j)} b_{j'} \\
&= \sum_{r_{j'} \in \mathcal{T}(j)} b_{j'}(n-1) \\
&\leq \sum_{r_{j'} \in \mathcal{T}(j)} b_{j'} \cdot \frac{\rho_{max}}{\rho_{min}} \left(\omega_{v^*}(j') + \sum_{e \in P_{OPT}(s_{j'}, t_{j'}; v^*)} \omega_e(j') \right) \\
&= \sum_{r_{j'} \in \mathcal{T}(j)} b_{j'} \cdot \frac{\rho_{max}}{\rho_{min}} \left(\sum_{e \in P_{OPT}(s_{j'}, t_{j'}; v^*)} \frac{c_e(j')}{B_e} + \frac{c_{v^*}(j')}{\rho_{max} \cdot C_{v^*}} \right) \\
&\leq \frac{\rho_{max}}{\rho_{min}} \sum_{r_{j'} \in \mathcal{T}(j)} b_{j'} \left(\sum_{e \in P_{OPT}(s_{j'}, t_{j'}; v^*)} \frac{c_e(j)}{B_e} + \frac{c_{v^*}(j)}{\rho_{max} \cdot C_{v^*}} \right)
\end{aligned} \tag{40}
$$

$$
\begin{aligned}
&= \frac{\rho_{max}}{\rho_{min}} \sum_{r_{j'} \in \mathcal{T}(j)} \left(\sum_{e \in P_{OPT}(s_{j'}, t_{j'}; v^*)} \frac{c_e(j) \cdot b_{j'}}{B_e} + \frac{c_{v^*}(j) \cdot b_{j'}}{\rho_{max} \cdot C_{v^*}} \right) \\
&= \frac{\rho_{max}}{\rho_{min}} \left(\sum_{e \in P_{OPT}(s_{j'}, t_{j'}; v^*)} c_e(j) \frac{\sum_{r_{j'} \in \mathcal{T}(j)} b_{j'}}{B_e} \right. \\
&\qquad \left. + c_{v^*}(j) \frac{\sum_{r_{j'} \in \mathcal{T}(j)} b_{j'}}{\rho_{max} C_{v^*}} \right) \\
&\leq \frac{\rho_{max}}{\rho_{min}} \sum_{e \in P_{OPT}(s_{j'}, t_{j'}; v^*)} c_e(j) + \frac{\gamma c_{v^*}(j)}{\rho_{min}} \quad \text{let } \gamma = \max_{j'=1}^{j} \{b_{j'}\},
\end{aligned} \tag{41}
$$

$$
\begin{aligned}
&\leq \frac{\rho_{max}}{\rho_{min}} \sum_{e \in E} c_e(j) + \frac{1}{\rho_{min}} \sum_{v \in V} c_v(j) \cdot \gamma \\
&\leq 2(n-1) \cdot \frac{\rho_{max}}{\rho_{min}} (\gamma \cdot \mathbb{C}(j) \log \alpha + \mathbb{B}(j) \log \beta).
\end{aligned} \tag{42}
$$

Inequality (40) holds since the utilization ratio of both computing and bandwidth resources do not decrease and thus the usage weight of each node or edge in G_j does not decrease with more and more request admissions.

Let γ be the maximum bandwidth demanded by all requests, i.e., $\gamma = \max\{b_{j'} \mid 1 \leq j' \leq j\}$, and let δ be the maximum computing resource demanded by any request, i.e., $\delta = \max\{\phi_{j'} \cdot C(SFC^{(k')}) \mid 1 \leq j' \leq j\}$. Inequality (41) holds since the accumulated bandwidth consumption of all admitted requests whose service function chains are implemented in any data center $v \in V$ is no greater than $C_v \cdot \gamma$ by the assumption that $C(SFC^{(k)}) > 1$ for any type k of service function chain, and data center v is able to admit at most C_v requests, i.e., $\sum_{r_{j'} \in \mathcal{T}(j)} \sum_{e \in P_{OPT}(s_{j'}, t_{j'}; v^*)} b_{j'} \leq \gamma \cdot C_{v^*}$. Meanwhile, all algorithms for the problem, including the optimal offline algorithm OPT, the accumulated usage of bandwidth in any link is no greater than the link bandwidth capacity, i.e., $\sum_{r_{j'} \in \mathcal{T}(j)} \sum_{e \in P_{OPT}(s_{j'}, t_{j'}; v^*)} b_{j'} \leq B_e$.

By Inequality (42), we have

$$
\mathbb{B}_{OPT}(j) - \mathbb{B}(j) \leq 2 \cdot \frac{\rho_{max}}{\rho_{min}} (\gamma \cdot \mathbb{C}(j) \log \alpha + \mathbb{B}(j) \log \beta). \tag{43}
$$

The following inequality can be shown by adopting a similar method as above

$$
\mathbb{C}_{OPT}(j) - \mathbb{C}(j) \leq 2 \cdot \frac{\rho_{max}}{\rho_{min}} (\mathbb{C}(j) \log \alpha + \delta \cdot \mathbb{B}(j) \log \beta). \tag{44}
$$

Recall that $\mathcal{A}(j)$ is the set of requests admitted by the online algorithm, Algorithm 3, and $\mathcal{T}(j)$ is the set of requests rejected by Algorithm 3 but admitted by algorithm OPT prior to the arrival of request r_j.

Let $p(\mathcal{A}(j))$ and $p(\mathcal{T}(j))$ be the total profit for the admissions of requests in $\mathcal{A}(j)$ and $\mathcal{T}(j)$, respectively. We here abuse the notation OPT to denote it as both the optimal offline algorithm and the solution delivered by the optimal algorithm. It is clear that the profit collected by admitting a request is always non-negative as this is guaranteed by the admission control policy. Without loss of generality, for any server s at a data center v, we assume that the amount of its energy consumption when it is in idle status to its total energy consumption is no greater than a constant ξ with $0 < \xi < 1$, i.e., $\frac{\mathcal{E}_v^{idle}}{\mathcal{E}_v^{idle} + \eta_v \cdot \mathcal{U}_s} \leq \xi$. Thus, for each data center $v \in V$, we have

$$
\frac{\sum_{s \in Q_v} \mathcal{E}_v^{idle}}{\sum_{s \in Q_v} \mathcal{E}_v^{idle} + \sum_{s \in Q_v} \eta_v \cdot \mathcal{U}_s} \leq \xi. \tag{45}
$$

The competitive ratio of Algorithm 3 then is where $x = \frac{\mathbb{C}(j)}{\mathbb{B}(j)}$, $\alpha = \beta = 2n$, $a_1 = \max_{v \in V} \{\frac{\rho_v \cdot |\mathcal{Q}_v| \cdot \eta_v}{C_v}\}$, and $a_2 = \frac{a_1}{1-\xi}$. Notice that a_1, a_2, and ξ are constants.

$$
\begin{aligned}
\frac{p(\mathcal{A}(j))}{OPT} &\geq \frac{p(\mathcal{A}(j))}{p(\mathcal{A}(j))+p(\mathcal{T}(j))} = \frac{1}{\frac{p(\mathcal{T}(j))}{p(\mathcal{A}(j))}+1} \\
&= \frac{1}{\frac{\lambda_1\cdot\sum_{r_{j'}\in\mathcal{T}(j)}\phi_{j'}\cdot C(SFC^{(k')})+\lambda_2\cdot\sum_{r_{j'}\in\mathcal{T}(j)}b_{j'}-\sum_{v\in V}\rho_v\mathcal{E}_v}{\lambda_1\mathbb{C}(j)+\lambda_2\mathbb{B}(j)-\sum_{v\in V}\rho_v\mathcal{E}_v}+1}, \qquad \text{by Eq. (6)} \\
&\geq \frac{1}{\frac{\lambda_1\cdot(\mathbb{C}_{OPT}(j)-\mathbb{C}(j))+\lambda_2\cdot(\mathbb{B}_{OPT}(j)-\mathbb{B}(j))-\sum_{v\in V}\rho_v\mathcal{E}_v}{\lambda_1\mathbb{C}(j)+\lambda_2\mathbb{B}(j)-\sum_{v\in V}\rho_v\mathcal{E}_v}+1} \\
&\geq \frac{1}{\frac{2\cdot\frac{\rho_{max}}{\rho_{min}}\cdot\lambda_1(\mathbb{C}(j)\log\alpha+\delta\mathbb{B}(j)\log\beta)+2\cdot\frac{\rho_{max}}{\rho_{min}}\cdot\lambda_2(\gamma\,\mathbb{C}(j)\log\alpha+\mathbb{B}(j)\log\beta)-\sum_{v\in V}\rho_v\mathcal{E}_v}{\lambda_1\mathbb{C}(j)+\lambda_2\mathbb{B}(j)-\sum_{v\in V}\rho_v\mathcal{E}_v}+1}, \qquad \text{by Ineqs. (43) and (44)} \\
&= \frac{1}{\frac{2\cdot\frac{\rho_{max}}{\rho_{min}}\cdot\lambda_1(\mathbb{C}(j)\log\alpha+\delta\mathbb{B}(j)\log\beta)+2\cdot\frac{\rho_{max}}{\rho_{min}}\cdot\lambda_2(\gamma\mathbb{C}(j)\log\alpha+\mathbb{B}(j)\log\beta)-\sum_{v\in V}\rho_v\mathcal{E}_v}{\lambda_1\mathbb{C}(j)+\lambda_2\mathbb{B}(j)-\sum_{v\in V}\rho_v|\mathcal{Q}_v|\mathcal{E}_v^{idle}-\sum_{v\in V}\rho_v|\mathcal{Q}_v|\eta_v\cdot\mathcal{U}_v}+1}, \qquad \text{by Eq. (4)} \\
&\geq \frac{1}{\frac{2\cdot\frac{\rho_{max}}{\rho_{min}}\cdot\lambda_1(\mathbb{C}(j)\log\alpha+\delta\,\mathbb{B}(j)\log\beta)+2\cdot\frac{\rho_{max}}{\rho_{min}}\cdot\lambda_2(\gamma\mathbb{C}(j)\log\alpha+\mathbb{B}(j)\log\beta)}{\lambda_1\mathbb{C}(j)+\lambda_2\mathbb{B}(j)-\sum_{v\in V}\rho_v|\mathcal{Q}_v|\mathcal{E}_v^{idle}-\sum_{v\in V}\rho_v|\mathcal{Q}_v|\eta_v\cdot\mathcal{U}_v}+1} \qquad (46) \\
&\geq \frac{1}{\frac{2\cdot\frac{\rho_{max}}{\rho_{min}}\cdot\lambda_1(\mathbb{C}(j)\log\alpha+\delta\mathbb{B}(j)\log\beta)+2\cdot\frac{\rho_{max}}{\rho_{min}}\cdot\lambda_2(\gamma\mathbb{C}(j)\log\alpha+\mathbb{B}(j)\log\beta)}{\lambda_1\mathbb{C}(j)+\lambda_2\mathbb{B}(j)-(\frac{\xi}{1-\xi}+1)\cdot a_1\mathbb{C}(j)}+1}, \quad \text{by Ineq. (45) and let } a_1=\max_{v\in V}\{\frac{\rho_v\cdot|\mathcal{Q}_v|\cdot\eta_v}{C_v}\} \\
&= \frac{1}{\frac{2\cdot\frac{\rho_{max}}{\rho_{min}}(\lambda_1(\frac{\mathbb{C}(j)}{\mathbb{B}(j)}\log\alpha+\delta\log\beta)+\lambda_2(\gamma\frac{\mathbb{C}(j)}{\mathbb{B}(j)}\log\alpha+\log\beta))}{\lambda_1\frac{\mathbb{C}(j)}{\mathbb{B}(j)}+\lambda_2-\frac{a_1\,\mathbb{C}(j)}{1-\xi\mathbb{B}(j)}}+1} \\
&= \frac{1}{\frac{2\cdot\frac{\rho_{max}}{\rho_{min}}(\lambda_1(\frac{\mathbb{C}(j)}{\mathbb{B}(j)}\log\alpha+\delta\log\beta)+\lambda_2(\gamma\frac{\mathbb{C}(j)}{\mathbb{B}(j)}\log\alpha+\log\beta))}{\lambda_1\frac{\mathbb{C}(j)}{\mathbb{B}(j)}+\lambda_2-a_2\frac{\mathbb{C}(j)}{\mathbb{B}(j)}}+1}, \quad \text{let } a_2=\frac{a_1}{1-\xi} \\
&= \frac{1}{2\log 2n\cdot\frac{\rho_{max}}{\rho_{min}}\frac{(\lambda_1+\lambda_2\gamma)x+\lambda_2+\lambda_1\delta}{(\lambda_1-a_2)x+\lambda_2}+1}, \quad \text{when } \alpha=\beta=2n \text{ and let } x=\frac{\mathbb{C}(j)}{\mathbb{B}(j)} \\
&= \frac{1}{2\log 2n\cdot\frac{\rho_{max}}{\rho_{min}}(1+\frac{(\lambda_2\gamma+a_2)x+\lambda_1\delta}{(\lambda_1-a_2)x+\lambda_2})+1},
\end{aligned}
$$

Since the profit collected is always non-negative, the term $(\lambda_1-a_2)x+\lambda_2$ in Inequality (46) is always non-negative too. The rest is to estimate the upper bound on term $2\log 2n\cdot\frac{\rho_{max}}{\rho_{min}}(1+\frac{(\lambda_2\gamma+a_2)x+\lambda_1\delta}{(\lambda_1-a_2)x+\lambda_2})+1$ in the denominator of Inequality (46). Let $f(x)=\frac{(\lambda_2\gamma+a_2)x+\lambda_1\delta}{(\lambda_1-a_2)x+\lambda_2}$, $x\in(0,+\infty)$. The first derivative of $f(x)$ is

$$f'(x)=\frac{(\lambda_2\gamma+a_2)\lambda_2-(\lambda_1-a_2)\lambda_1\delta}{\left((\lambda_1-a_2)x+\lambda_2\right)^2}, \qquad (47)$$

and the term $\frac{1}{((\lambda_1-a_2)x+\lambda_2)^2}$ in $f'(x)$ always is non-negative. In the following we show that function $f(x)$ is upper bounded. We distinguish our discussion into the following three cases. Case 1. If $\frac{\lambda_2\gamma+a_2}{\lambda_1-a_2}-\frac{\lambda_1\delta}{\lambda_2}>0$, then $f'(x)>0$ and $\lim_{x\to+\infty}f'(x)=0$. Function $f(x)$ thus is an increasing function and $\lim_{x\to+\infty}f(x)=\frac{\lambda_2\gamma+a_2}{\lambda_1-a_2}$. Case 2. If $\frac{\lambda_2\gamma+a_2}{\lambda_1-a_2}-\frac{\lambda_1\delta}{\lambda_2}<0$, then $f'(x)<0$ and $\lim_{x\to+\infty}f'(x)=0$. Thus, $f(x)$ is a decreasing function and $\lim_{x\to 0}f(x)=\frac{\lambda_1\delta}{\lambda_2}$. Case 3. If $\frac{\lambda_2\gamma+a_2}{\lambda_1-a_2}-\frac{\lambda_1\delta}{\lambda_2}=0$, then $f'(x)=0$ and $f(x)=\frac{\lambda_2\gamma+a_2}{\lambda_1-a_2}=\frac{\lambda_1\delta}{\lambda_2}$.

In summary, the upper bound of $f(x)$ is $\frac{\lambda_2\gamma+a_2}{\lambda_1-a_2}$ if $\frac{\lambda_2\gamma+a_2}{\lambda_1-a_2}-\frac{\lambda_1\delta}{\lambda_2}\geq 0$; $\frac{\lambda_1\delta}{\lambda_2}$ otherwise. Therefore, we have

$$\frac{p(\mathcal{A}(j))}{OPT}\geq\begin{cases}\frac{1}{2\log 2n\cdot\frac{\rho_{max}}{\rho_{min}}(1+\frac{\lambda_2\gamma+a_2}{\lambda_1-a_2})+1} & \text{if } \frac{\lambda_2\gamma+a_2}{\lambda_1-a_2}-\frac{\lambda_1\delta}{\lambda_2}\geq 0,\\ \frac{1}{2\log 2n\cdot\frac{\rho_{max}}{\rho_{min}}(1+\frac{\lambda_1\delta}{\lambda_2})+1} & \text{otherwise.}\end{cases} \qquad (48)$$

Notice that λ_1, λ_2, a_2, ρ_{max}, ρ_{min}, δ, and γ are constants. Thus, $p(\mathcal{A}(j))\geq\frac{OPT}{\theta_1\cdot\log n+\theta_2}$, where θ_1 and θ_2 are non-negative constants, when $\alpha=\beta=2n$.

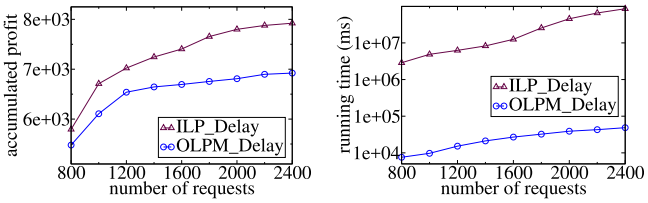
The time complexity analysis of `Algorithm 3` is omitted, due to space limitation. □

7 Performance Evaluation

In this section, we evaluate the performance of the proposed algorithms through experimental simulations. We also investigate the impact of important parameters on the performance of the proposed algorithms.

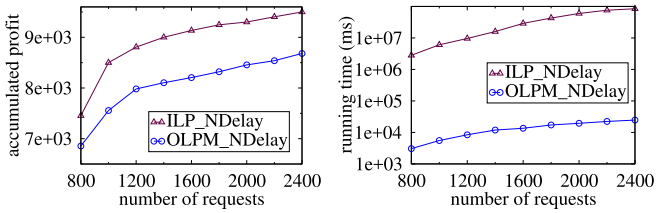
7.1 Experiment Settings

We consider an SDN $G=(S\cup V,E)$ consisting of from 10 to 250 nodes. We adopt a commonly used tool GT-ITM [8] to generate network topologies. The number of data centers in each network is set at 10 percent of network size. The computing capacity of each data center varies from



(a) The accumulated profit (b) The running time

Fig. 3. Performance of different algorithms with 3 data centers and with delay constraints.



(a) The accumulated profit (b) The running time

Fig. 4. Performance of different algorithms with 3 data centers and without delay constraints.

4,000 to 8,000 GHz [32]. The number of servers in each data center is randomly drawn between 1,000 and 2,000. The energy consumption of an idle server per second is drawn between 100 and 200 $watt$ [26] which is a constant; otherwise, its energy consumption is determined by its energy consumption rate η that is drawn within $[0.5, 1.5]$ [20]. For each data center, the unit energy price per hour in its location is drawn from $[15, 50]$ per MWh [25], [31]. The bandwidth capacity of each link varies between 2,000 and 20,000 Mbps [15], and the transmission delay on a link varies from 2 to 5 ms [15]. The revenue accumulation weights λ_1 and λ_2 are two constants randomly drawn within $[0.05, 0.12]$ and $[0.15, 0.22]$ respectively, following typical charges in Amazon EC2 [1]. The number of different types of service function chains K is set at 10. The bandwidth requirement of each type of service function chain instance is set between 10 and 20 Mbps [6], and their computing demand is set from 2 to 4 GHz [6], [19]. The processing rate of each service function chain instance is randomly drawn from 2 to 20 basic data packets per millisecond [19]. For each generated request r_j, two switch nodes in S are randomly selected as its source s_j and destination t_j. The packet rate ϕ_j of each request r_j is randomly drawn from 1 to 10 packets/second [16]. Its end-to-end packet delay constraint D_j varies from 10 to 100 ms [19], and its type of service function chain $SFC^{(k)}$ is randomly chosen from the K types. The value in each figure is the mean of the results out of 50 SDN instances of the same size. The running time of an algorithm is obtained on a machine with 3.4 GHz Intel i7 Quad-core CPU and 16 GB RAM. Unless otherwise specified, these parameters will be adopted in the default setting.

7.2 Performance Evaluation of Online Algorithms

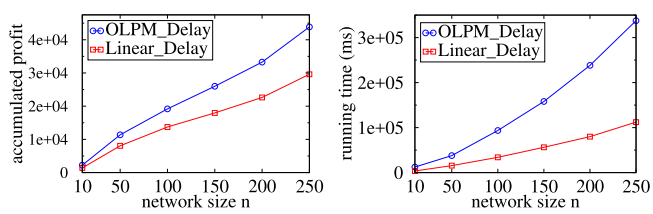
We refer to online algorithms `Algorithms` 2 and 3 for the problem with and without end-to-end delays as algorithms `OLPM_Delay` and `OLPM_NDelay`, respectively. We first evaluate the performance of these two algorithms against the optimal ILP solutions with and without delay constraints that are termed as algorithms `ILP_Delay` and `ILP_NDelay`, respectively. We then evaluate the online algorithms against two baseline heuristics: `Linear_Delay` and `Linear_NDelay` which represent with and without the end-to-end delay constraints, where for each request r_j, algorithms `Linear_Delay` and `Linear_NDelay` first remove all nodes and links from G that do not have enough residual resources to admit r_j, and then assign each data center node and each link in the resulting subgraph with a weight of one. They finally find a (delay-constrained) shortest path in the resulting subgraph from s_j to t_j via a data center $v \in V$ without (with) the end-to-end delay constraint on the request.

We now evaluate the performance of algorithms `OLPM_Delay` and `OLPM_NDelay` against the optimal solutions of algorithms `ILP_Delay` and `ILP_NDelay`, by varying the number of requests from 800 to 2,400 while fixing the number of data centers at 3. Fig. 3 depicts the total profits and running times of different algorithms.

We can see from Fig. 3a that algorithm `OLPM_Delay` achieves a near optimal profit, i.e., it achieves at least 88.3 percent of the performance compared with the optimal solution delivered by algorithm `ILP_Delay`. For example, algorithm `OLPM_Delay` can achieve 93.1 percent of the optimal profit delivered by algorithm `ILP_Delay` when the number of requests is no more than 1,200; and it achieves 88.3 percent of the optimal profit for 2,400 requests. It can be seen from Fig. 3b that algorithm `ILP_Delay` takes a prohibitively long time, while algorithm `OLPM_Delay` only takes a small fraction of the running time of algorithm `ILP_Delay`. Notice that with the increase on the number of requests, algorithm `ILP_Delay` fails to deliver a solution within a reasonable amount of time when the number of requests reaches 2,400. The similar performance behaviors can be observed between algorithms `OLPM_NDelay` and `ILP_NDelay` too, which are illustrated in Figs. 4a and 4b, respectively.

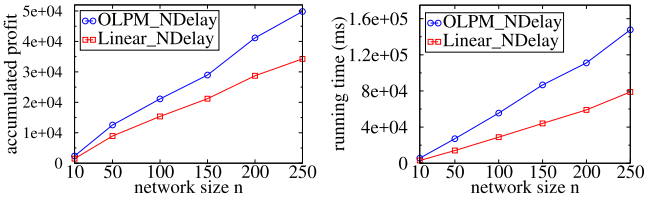
The rest is to evaluate the performance of algorithms `OLPM_Delay` and `OLPM_NDelay` against their corresponding benchmark heuristics `Linear_Delay` and `Linear_NDelay`, respectively, for a set of 10,000 requests, by varying the number of nodes from 10 to 250 while keeping the other parameters fixed, i.e., $\alpha = \beta = 2n$, $\sigma_1 = (n-1) \cdot \rho_{min}/\rho_{max}$, and $\sigma_2 = n - 1$.

Fig. 5 demonstrates the performance behaviors of the mentioned algorithms. From Figs. 5a and 6a, we can see that both algorithms `OLPM_Delay` and `OLPM_NDelay` outperform their benchmark counterparts: algorithms `Linear_Delay` and `Linear_NDelay`. Specifically, algorithm `OLPM_Delay` outperforms algorithm `Linear_Delay`, and



(a) The accumulated profit (b) The running time

Fig. 5. Performance of different online algorithms by varying network size from 10 to 250 and with delay constraints.



(a) The accumulated profit (b) The running time

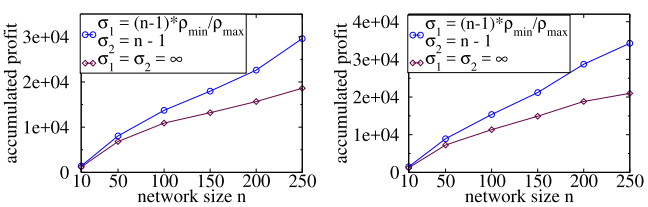
Fig. 6. Performance of different online algorithms by varying network size from 10 to 250 without delay constraints.

the performance gap between them becomes larger and larger with the increase in network size. As shown in Fig. 5a, algorithm `OLPM_Delay` delivers 48.2 percent more profits than that by algorithm `Linear_Delay` when the network size is set at 250. The similar performance can be observed by algorithm `OLPM_NDelay` against algorithm `Linear_NDelay` too. In particular, as shown in Fig. 6a, the accumulated profit delivered by algorithm `OLPM_NDelay` is 45.4 percent more than that by algorithm `Linear_NDelay` when $n = 250$.

The rationale is that both algorithms `OLPM_Delay` and `OLPM_NDelay` assign higher usage weights to over-utilized resources while assigning lower usage weights to under-utilized resources, thus the resources in the network can be fully utilized to meet the demands of user requests, thereby achieving higher profits. In contrast, both algorithms `Linear_Delay` and `Linear_NDelay` do not take into account the utilization of resources, thus the resource usage on some data centers and links are overloaded. Fig. 5b depicts the running time curves of algorithms `OLPM_Delay` and `Linear_Delay`, where algorithm `Linear_Delay` takes less time than algorithm `OLPM_Delay` in all network sizes. The similar results on the running time of algorithms `OLPM_NDelay` and `Linear_NDelay` can also be seen in Fig. 6b.

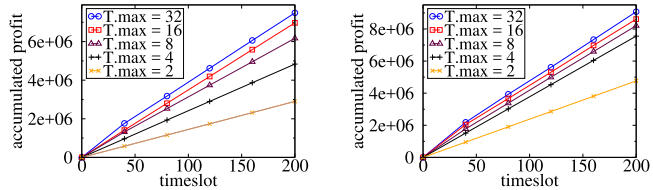
7.3 Parameter Impact on the Algorithm Performance

In the following, we first evaluate the impact of the admission control threshold parameters σ_1 and σ_2 on the performance of algorithms `OLPM_Delay` and `OLPM_NDelay`. Fig. 7 plots their performance curves with and without the admission control thresholds, from which it can be seen that both of them deliver less profits if no admission control policy is applied. With the increase in network size n, the performance gap of algorithm `OLPM_Delay` with and without the admission control thresholds becomes larger as shown in Fig. 7a. For instance, the ratio of the profit delivered by algorithm `OLPM_Delay` with and without the admission control grows from 1.18 to 1.59 when $n = 10$ and 250



(a) The accumulated profit by `OLPM_Delay` (b) The accumulated profit by `OLPM_NDelay`

Fig. 7. The accumulated profit delivered by `OLPM_Delay` and `OLPM_NDelay` with thresholds $\sigma_1 = (n-1) \cdot \rho_{min}/\rho_{max}$, $\sigma_2 = n - 1$, and without the thresholds $\sigma_1 = \sigma_2 = \infty$ when $\alpha = \beta = 2n$.



(a) The accumulated profit by algorithm `OLPM_Delay` (b) The accumulated profit by algorithm `OLPM_NDelay`

Fig. 8. The accumulated profit of online algorithms `OLPM_Delay`, and `OLPM_NDelay` with different maximum durations of requests when the network size is 200.

respectively. This is due to that the distance between the source and the destination of a request can be very long in a large network, thus consuming much more bandwidth resource for routing its data traffic. Under the admission control policy, algorithm `OLPM_Delay` is able to reject those requests beyond the given threshold, thereby enabling to admit future requests and achieving higher profits. The performance gap of algorithm `OLPM_NDelay` with and without the admission control is similar to the one of algorithm `OLPM_Delay` with and without the admission control, as shown in Fig. 7b.

We then evaluate the impact of the maximum duration T_{max} of request implementation on the performance of algorithms `OLPM_Delay` and `OLPM_NDelay` in a network with 200 nodes, by varying T_{max} from 2 to 32. Assume that the finite time horizon is slotted into equal time slots. The request admissions and resource releases are performed in the beginning of each time slot. The duration of each request implementation is randomly drawn in $[1, T_{max}]$. The profit collected by admitting a request r_j is the duration of request implementation in a data center times its profit $p(r_j)$ in a single time slot. We consider a time horizon consisting of 200 time slots with up to 1,000 requests per time slot. From Fig. 8a, it can be seen that the longer the maximum duration, the more profits delivered by algorithm `OLPM_Delay`. The similar performance of algorithm `OLPM_NDelay` can be observed in Fig. 8b.

8 Conclusion

In this paper, we investigated the dynamic profit maximization problem in distributed clouds by dynamically admitting delay-aware requests with network function service requirements, assuming that each request is admitted on a pay-as-you-go basis, subject to various network resource capacity constraints. We first proposed an ILP solution for the offline version of the problem. We then developed an online heuristic for the problem. We also devised an online algorithm with a provable competitive ratio for a special case of the problem where the end-to-end delay constraint of each request can be negligible. We finally evaluated the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms are promising.

Acknowledgments

We really appreciate the six anonymous referees and the associate editor for their expertise comments and constructive suggestions, which have helped us improve the quality

and presentation of the paper greatly. It is acknowledged that the work by Z. Xu is supported by National Natural Science Foundation of China (Grant No. 61802048), the fundamental research funds for the central universities in China (Grant No. DUT17RC(3)061), and the Xinghai Scholar Program in Dalian University of Technology, China, and the work by S. Guo is supported by National Natural Science Foundation of China (Grant No. 61872310).

REFERENCES

[1] Amazon Web Services, Inc., "Amazon EC2 instance configuration," 22 Mar., 2018. [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-ug.eps

[2] P. Borylo, A. Lason, J. Rzasa, A. Szymanski, and A. Jajszczyk, "Energy-aware fog and cloud interplay supported by wide area software defined networking," in *Proc. IEEE Int. Conf. Commun.*, 2016, pp. 1–7.

[3] M. Charikar, Y. Naamad, J. Rexford, and X. K. Zou, "Multi-commodity flow with in-network processing," (2014). [Online]. Available: http://www.cs.princeton.edu/ jrex/papers/mopt14.pdf

[4] N. Chowdhury and R. Boutaba, "Network virtualization: State of the art and research challenges," *IEEE Commun. Mag.*, vol. 47, no. 7, pp. 20–26, Jul. 2009.

[5] S. D'Oro, S. Palazzo, and G. Schembra, "Orchestrating softwarized networks with a marketplace approach," in *Proc. Int. Conf. Future Netw. Commun.*, 2017, pp. 352–360.

[6] M. Ghaznavi, N. Shahriar, R. Ahmed, and R. Boutaba, "Service function chaining simplified," (Feb. 2016). [Online]. Available: http://arxiv.org/abs/1601.00751

[7] C. Ghribi, M. Mechtri, and D. Zeghlache, "A dynamic programming algorithm for joint VNF placement and chaining," in *Proc. ACM Workshop Cloud-Assisted Netw.*, Dec. 2016, pp. 19–24.

[8] GT-ITM is a network topology generating tool, (2018). [Online]. Available: https://www.cc.gatech.edu/projects/gtitm/

[9] S. Gu, Z. Li, C. Wu, and C. Huang, "An efficient auction mechanism for service chains in the NFV market," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 10–15.

[10] A. Gushchin, A. Walid, and A. Tang, "Scalable routing in SDN-enabled networks with consolidated middleboxes," in *Proc. ACM SIGCOMM Workshop Hot Topics Middleboxes Netw. Function Virtualization*, 2015, pp. 55–60.

[11] M. Huang, W. Liang, Z. Xu, and S. Guo, "Efficient algorithms for throughput maximization in software-defined networks with consolidated middleboxes," *IEEE Trans. Netw. Service Manage.*, vol. 14, no. 3, pp. 631–645, Sep. 2017.

[12] M. Jia, W. Liang, and Z. Xu, "QoS-aware task offloading in distributed cloudlets with virtual network function services," in *Proc. 20th ACM Int. Conf. Model. Anal. Simul. Wireless Mobile Syst.*, 2017, pp. 109–116.

[13] Y. Jia, C. Wu, Z. Li, F. Le, and A. Liu, "Online scaling of NFV service chains across geo-distributed datacenters," *IEEE Trans. Netw.*, vol. 26, no. 2, pp. 699–710, Apr. 2018.

[14] A. Juttner, B. Szviatovski, I. Mecs, and Z. Rajko, "Lagrange relaxation based method for the QoS routing problem," in *Proc. IEEE INFOCOM*, 2001, pp. 859–868.

[15] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *J. Sel. Areas Commun.*, vol. 29, pp. 1765–1775, 2011.

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

[17] T. Lukovszki and S. Schmid, "Online admission control and embedding of service chains," in *Proc. Int. Colloq. Structural Inf. Commun. Complexity*, 2014, pp. 104–118.

[18] Y. Ma, W. Liang, and Z. Xu, "Online revenue maximization in NFV-enabled SDNs," in *Proc. IEEE Int. Conf. Commun.*, 2018, pp. 1–7.

[19] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici, "ClickOS and the art of network function virtualization," in *Proc. USENIX Symp. Netw. Syst. Des. Implementation*, 2014, pp. 459–473.

[20] L. Minas and B. Ellison, "The problem of power consumption in servers," 24 Mar. 2018. [Online]. Available: https://www.infoq.com/articles/power-consumption-servers, Report by Intel

[21] ONF White Paper, "Software-defined networking: The new norm for networks," Report by Opening Networking Foundation (ONF), 13 Apr., 2012.

[22] S. K. Panda and S. Mishra, "An efficient server minimization algorithm for internet distributed systems," *Int. J. Rough Sets Data Anal.*, vol. 4, no. 4, pp. 17–30, 2017.

[23] P. Quinn and T. Nadeau, "Problem statement for service function chaining," *Internet requests for comments, IETF*, http://www.rfc-editor.org/rfc/rfc7498.txt, RFC 7498, Fremont, CA, USA, Apr. 2015.

[24] W. Racheg, N. Ghrada, and M. F. Zhani, "Profit-driven resource provisioning in NFV-based environments," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–7.

[25] L. Rao, X. Liu, M. D. Ilic, and J. Liu, "Distributed coordination of Internet data centers under multiregional electricity markets," *Proc. IEEE*, vol. 100, no. 1, pp. 269–282, Jan. 2012.

[26] L. Rao, X. Liu, L. Xie, and W. Liu, "Minimizing electricity cost: Optimization of distributed Internet data centers in a multi-electricity-market environment," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[27] S. Ren, Y. He, and F. Xu, "Provably-efficient job scheduling for energy and fairness in geographically distributed data centers," in *Proc. IEEE 32nd Int. Conf. Distrib. Comput. Syst.*, 2012, pp. 22–31.

[28] S. V. Rossem, et al., "Deploying elastic routing capability in an SDN/NFV-enabled environment," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw.*, 2015, pp. 22–24.

[29] P. Rost, et al. "Network slicing to enable scalability and flexibility in 5G mobile networks," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 72–79, May 2017.

[30] P. Wang, J. Lan, X. Zhang, Y. Hu, and S. Chen, "Dynamic function composition for network service chain: Model and optimization," *Comput. Netw.*, vol. 92, pp. 408–418, 2015.

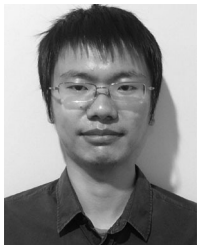
[31] Z. Xu and W. Liang, "Operational cost minimization of distributed data centers through the provision of fair request rate allocations while meeting different user SLAs," *Comput. Netw.*, vol. 83, pp. 59–75, 2015.

[32] Z. Xu, W. Liang, A. Galis, and Y. Ma, "Throughput maximization and resource optimization in NFV-enabled networks," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–7.

[33] H. Yuan, et al. "WARM: Workload-aware multi-application task scheduling for revenue maximization in SDN-based cloud data center," *IEEE Access*, vol. 6, pp. 645–657, 2018.

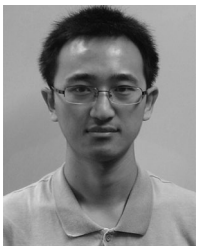
[34] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. C. M. Lau, "An online stochastic buy-sell mechanism for VNF chains in the NFV market," *IEEE J. Sel. Areas Commun. Special Issue Game Theory Netw.*, vol. 35, no. 2, pp. 1–15, Feb. 2017.

[35] Q. Zhang, M. F. Zhani, R. Boutaba, and J. L. Hellerstein, "Dynamic heterogeneity-aware resource provisioning in the cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 14–28, Jan.–Mar. 2014.

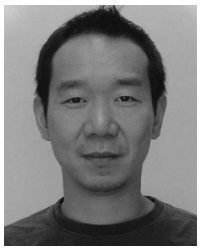


Yu Ma received the BSc (first class Honours) degree in computer science from the Australian National University, in 2015. He is currently working toward the PhD degree in the Research School of Computer Science, Australian National University. His research interests include software defined networking, Internet of Things (IoT), and social networking.

Weifa Liang (M'99–SM'01) received the BSc degree in computer science from Wuhan University, China, in 1984, the ME degree in computer science from the University of Science and Technology of China, in 1989, and the PhD degree in computer science from the Australian National University, in 1998. He is currently a full professor with the Research School of Computer Science, 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 and a member of the ACM.



Zichuan Xu (M'17) received the BSc and ME degrees in computer science from the Dalian University of Technology, China, in 2008 and 2011, respectively, and the PhD degree in computer science from the Australian National University, in 2016. He was a research associate with the Department of Electronic and Electrical Engineering, University College London, United Kingdom. He currently is an associate professor with the School of Software, Dalian University of Technology, China. His research interests include cloud computing, software-defined networking, network function virtualization, wireless sensor networks, routing protocol design for wireless networks, algorithmic game theory, and optimization problems. He is a member of the IEEE.



Song Guo (M'02-SM'11) received the PhD degree in computer science from the University of Ottawa. He is a full professor with the Department of Computing, Hong Kong Polytechnic University. He was a professor with the University of Aizu. His research interests are mainly in the areas of big data, cloud computing and networking, and distributed systems with more than 400 papers published in major conferences and journals. His work was recognized by the 2016 Annual Best of Computing: Notable Books and Articles in Computing in the *ACM Computing Reviews*. He is the recipient of the 2017 IEEE Systems Journal Annual Best Paper Award and other five Best Paper Awards from IEEE/ACM conferences. He was an associate editor of the *IEEE Transactions on Parallel and Distributed Systems* and an IEEE ComSoc distinguished lecturer. He is now on the editorial board of the *IEEE Transactions on Emerging Topics in Computing*, the *IEEE Transactions on Sustainable Computing*, the *IEEE Transactions on Green Communications and Networking*, and the *IEEE Communications*. He also served as general, TPC and symposium chair for numerous IEEE conferences. He currently serves as a director and member of the Board of Governors of ComSoc. He is a senior member of the IEEE.