Affinity-Aware VNF Placement in Mobile Edge Clouds via Leveraging GPUs

Zichuan Xu, *Member, IEEE*, Zhiheng Zhang, John C. S. Lui, *Fellow, IEEE*, Weifa Liang, *Senior Member, IEEE*, Qiufen Xia, *Member, IEEE*, Pan Zhou, *Senior Member, IEEE*, Wenzheng Xu, *Member, IEEE*, and Guowei Wu

Abstract—Mobile edge computing becomes a promising technology to mitigate the latency of various cloud services. In addition, network function virtualization (NFV) has been shown a great potential in reducing the operational cost of cloud services while enhancing the flexibility of virtual network function deployments, by implementing dedicated hardware network functions as pieces of software in generic servers. Recently, the GPU acceleration has been investigated to speed up flow processing in virtual network functions (VNFs), by leveraging the parallelism of GPUs. VNFs that need accelerations prefer to stay at cloudlets (locations) equipped with GPUs. However, little attention has been paid for the VNF placement that takes into account GPU-affinity in cloudlets of mobile edge clouds. In this paper, we consider the affinity-aware throughput maximization problem in a mobile edge cloud via leveraging the parallelism on GPUs for user requests with VNF requirements. We consider two types of affinities in the VNF placement: The *soft-affinity* that allows VNFs to be executed by either CPUs or GPUs in cloudlets; and the *hard-affinity* that only allows VNFs to be placed to the GPUs of a specified set of cloudlets. We formulate two corresponding VNF placement problems in a mobile edge cloud. Specifically, we first propose an exact solution to the soft-affinity throughput maximization problem by formulating an Integer Linear Program (ILP). We then propose an efficient algorithm for the problem, by proposing a randomized algorithm with a provable approximation ratio for the hard-affinity-aware throughput maximization problem and extending the proposed approximation algorithm to the soft-affinity throughput maximization problem. Furthermore, assuming that user requests arrive into the mobile edge cloud one by one without the knowledge of future arrivals, we devise an online algorithm with a good competitive ratio for this dynamic hard-affinity-aware throughput maximization problem. Finally, we evaluate the performance of the proposed algorithms, through simulations and implementations in a real test-bed. Experimental results show that the performance of the proposed algorithms outperform their existing counterparts and achieve higher throughput.

Index Terms—VNF placement, GPU affinity, mobile edge clouds, approximation and online algorithms

1 Introduction

Mobile edge computing and Network function virtualization (NFV) promise to provide flexible and low-latency cloud services. Each of such services can be realized as virtualized network functions (VNFs) running as pieces of software in Virtual Machines (VMs) [13], [18], [24], [33] on generic servers. Recently, new advances in NFVs using GPU acceleration in cloudlets (servers with GPUs) further improve the processing speeds of the VNFs. However, not every cloudlet in a mobile edge cloud has GPU hardware installed, and VNFs that require packet processing accelerations need to reside in the cloudlets with GPUs [21]. This means that the VNF placement needs to take into account the *affinity* requirement of specific hardware such as GPUs. In this paper, we consider the problem of affinity-aware VNF placement in a mobile edge cloud, by exploring the parallelism of GPUs in some cloudlets in the mobile edge cloud.

Exploring the parallelism of GPUs in cloudlets to maximize the system throughput poses several challenges. First, a mobile edge cloud has various capacity constraints on its different types of resources. For example, the GPU resource in each cloudlet of the mobile edge cloud is precious and thus can only execute a limited number of GPU threads. Selecting the right user requests to accelerate their VNF running plays a vital role in enhancing the system performance. Also, each switch node is equipped with a Ternary Content-Addressable Memory (TCAM) to implement its forwarding routing table that contains a limited number of forwarding rules [8]. Such resource capacities of GPU, CPU, and TCAMs in switches need to be jointly considered. Otherwise, the imbalanced usage of resources may lead to a significant reduction in the system throughput and violations

- *Zichuan Xu, Zhiheng Zhang, and Guowei Wu are with the School of Software, Dalian University of Technology, and the Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian 116621, China. E-mail: {z.xu, wgwdut}@dlut.edu.cn, zhangzhiheng@mail.dlut.edu.cn.*
- *John C. S. Lui is with the Department of Computer Science & Engineering, The Chinese University of Hong Kong, Shatin, N.T, Hong Kong E-mail: cslui@cse.cuhk.edu.hk.*
- *Weifa Liang is with the Research School of Computer Science, Australian National University, Canberra, ACT 2601, Australia E-mail: wliang@cs.anu.edu.au.*
- *Qiufen Xia is with the International School of Information Science and Engineering, Dalian University of Technology, and the Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian 116621, China. E-mail: qiufenxia@dlut.edu.cn.*
- *Pan Zhou is with the Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China. E-mail: panzhou@hust.edu.cn.*
- *Wenzheng Xu is with the Department of computer network and communication, Sichuan University, Chengdu, Sichuan 610000, China E-mail: wenzheng.xu@scu.edu.cn.*

Manuscript received 3 Mar. 2020; revised 3 July 2020; accepted 13 Nov. 2020. Date of publication 1 Dec. 2020; date of current version 8 Nov. 2021. (Corresponding authors: Wenzheng Xu and Guowei Wu.) Recommended for acceptance by T. Li. Digital Object Identifier no. 10.1109/TC.2020.3041629

of delay requirements of users. Second, user requests with VNF requirements have both GPU affinity and end-to-end delay requirements in the mobile edge cloud, meeting one of the requirements may lead to the violation of the other one. For example, to maximize the system throughput, a user request with high demand of data volume may be placed to the cloudlets without GPUs, due to the sparsity of GPU resources. This however may violate its delay requirement due to the slow data processing of VNFs running on CPUs. Therefore, it is challenging to explore non-trivial interplays among GPU affinity, the delay and system throughput in a mobile edge cloud. Third, the arrivals of user requests with different VNF requirements are usually unknown in advance. How to dynamically admit user requests while considering their VNF affinities is challenging too.

Most existing studies [6], [9], [15], [16], [17], [20], [22], [40] did not incorporate the GPU affinity to the VNF placement in mobile edge clouds with cloudlets having both CPU and GPU capacities. In addition, most existing studies proposed heuristics for their problems without any performance guarantees. In contrast, in this paper we propose approximation and online algorithms with provable approximation and competitive ratios for the affinity-aware VNF placement problems under static and dynamic request scenarios.

To the best of our knowledge, we are the first to explore the affinity-awareness VNF placement in a mobile edge cloud equipped with GPU accelerators, by leveraging the parallelism of GPUs. We consider both hard-affinity VNFs that have to be placed to a specified set of cloudlets with GPUs and the soft-affinity that allows VNFs to be implemented by CPUs in cloudlets. To this end, we jointly consider CPU and GPU capacity constraints on cloudlets and the TCAM capacity constraints on switches in the mobile edge cloud. We propose an approximation algorithm with a provable approximation ratio, an efficient heuristic and an online algorithm with a competitive ratio for the offline and online versions of the affinity-aware throughput maximization problem in the mobile edge cloud.

The main contributions of this paper are as follows.

- We propose an exact solution to the soft-affinity-aware throughput maximization problem, by formulating an Integer Linear Program (ILP) solution when the problem size is small.
- We devise a heuristic solution to the soft-affinity-aware throughput maximization problem, by first devising proposing a scalable, efficient approximation algorithm with an approximation ratio for a special case of the problem, and then extending the proposed approximation algorithm for the soft-affinity-aware throughput maximization problem.
- We propose an online algorithm with a competitive ratio for the online hard-affinity aware throughput maximization problem, by adopting the primal-dual-update optimization technique [3].
- We evaluate the performance of the proposed algorithms through experimental simulations and implementations in a real testbed. Experimental results show that the proposed algorithms outperform existing methods.

The remainder of this paper is arranged as follows. Section 2 will survey state-of-the-arts on this topic. Section 3 will introduce the system model, notations and problem definitions. Section 4 will formulate an ILP solution to the soft-affinity-aware throughput maximization problem. Sections 5 and 6 will propose approximation, heuristic and online algorithms for the problem under static and dynamic settings. Section 7 will evaluate the performance of the proposed algorithms, and Section 8 concludes the paper.

2 Related Work

There are extensive studies on SDN, NFV and mobile edge computing [6], [9], [15], [16], [17], [20], [22], [30], [34]. For example, Song *et al.* [32] investigated the task assignment problem in a mobile edge network with node and link capacity constraints as well as security requirements. Chen and Wu [6] developed algorithms for the NFV middlebox placement, by balancing the set-up and bandwidth consumption costs.

Although the virtualization of network functions reduces the cost of network management and increases the flexibility of network function deployment, the flow processing time of some computation-intensive VNFs may be dragged down by the additional virtualization layer in the software stack. To avoid such increase of VNFs' processing time, there are recent research efforts on using of GPUs to accelerate VNF executions [39], [43], [45]. For example, G-NET [43] is a NFV system that adopts a novel GPU virtualization technique to enable spatial GPU sharing among VNFs. By designing a hypervisor layer and adopting Hyper-Q technique, G-NET enables spatial GPU sharing among the VNFs. GEN [45] is a framework that accelerates VNF execution in GPUs. An infrastructure for VNF acceleration is proposed to enable elastic network function scaling and runtime modification of service function chains. However, these studies focus on either effective GPU sharing by VNFs or the acceleration of a single VNF. It must be mentioned that these studies are made the very first contributions towards enabling VNF execution and acceleration. We are motivated by these studies. In this paper, we aim to explore the use of GPUs in mobile edge clouds to accelerate VNF executions, such that the system throughput is maximized, while meeting the delay requirements of user requests.

Several efforts aim to provision NFV-enabled network services in mobile edge clouds [26], [40]. For example, Nam *et al.* [26] studied service chaining in mobile edge networks with the aim to minimize the average service time of traffic flow. Yu *et al.* [40] formulated a QoS-aware reliable traffic routing problem for service function chaining in mobile edge networks. They showed that the problem is NP-hard, and developed an approximation algorithm for the problem. Huang *et al.* [17] studied the online multicasting in software-defined networks with both node and link capacity constraints. Jia *et al.* [19] proposed a solution of offloading mobile services with NFV instances in a mobile edge cloud, assuming that all VNFs in a service chain can be consolidated into a single edge node. Carpio *et al.* [5] investigated the joint active and backup stateless VNF placement problem. Zhang *et al.* [44] studied the problem of joint service chaining and request scheduling problem, by proposing a priority-driven weighted algorithm to improve resource utilization and a heuristic algorithm to reduce the response latency. Fei *et al.* [10] developed efficient methods for the deployment service chain instances in a mobile edge network. In addition, several studies focused on the management of stateful VNFs [37] and hardware acceleration for VNFs [21], [43]. For example, Yang *et al.* [37] proposed a set of efficient methods to enable fault-tolerant VNF placement in a mobile edge cloud. Li *et al.* [21] proposed a novel CPU-FPGA co-design framework for NFV with both high performance and flexibility and implemented a dynamic hardware library based on FPGA. The locality and hardware affinity requirements, however, have not been considered in these mentioned studies, not to mention that the TCAM capacity of SDN switches and computing resources in cloudlets have ever been jointly considered. Bao *et al.* [2] studied the problem of parallelizing the execution of many VNFs for acceleration, by using a directed acyclic graph (DAG) to model the dependency of VNFs, such that the dependencies in the DAG are preserved while the delay is minimized.

3 Preliminaries

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

3.1 System Model

We consider a mobile edge cloud $(G = V \cup CL, E)$ deployed in a wireless metropolitan area within the proximity of users, where V is a set of switches, CL is a set of cloudlets, and E is a set of links that interconnect switches and cloudlets in the network. Each switch $v_i \in V$ is installed with a TCAM that supports fast parallel look-up of forwarding rules. Such a TCAM can store a limited number of forwarding entries and usually are very expensive [7]. Denote by $N(v)$ the number of forwarding entries in the TCAM of switch $v \in V$ [7]. Also, a switch with the minimum forwarding table size can install at least $\alpha \cdot |CL|$ forwarding rules, where α is a given integer constant with $\alpha > 1$. Each cloudlet $cl_m \in CL$ has limited CPU and GPU resources to implement network functions. Let $C(cl_m)$ be the CPU resource capacity of cl_m, in terms of the total number of data packets that can be processed. Denote by C_{min} the minimum CPU resource capacity among cloudlets in CL, i.e, $C_{min} = \min_{cl_m \in CL} C(cl_m)$. Without loss of generality, we assume that the cloudlet with computing capacity C_{min} is capable to accommodate any single request. Cloudlet cl_m may be installed with GPUs, and its capacity is denoted by $Th(cl_m)$ in terms of the maximum number of GPU threads it can run in parallel. Also, data transfers at each link $e \in E$ incur transmission delays, and let d_e be the delay of transmitting a unit of packet along link e. An example of the mobile edge network is shown in Fig. 1.

3.2 VNF and User Requests

Each user usually requires to process its data traffic in a sequence of network functions such as IDS, firewall, and load balancers, to guarantee the performance and security of its data transfer. Such a requirement of a user request then can be expressed in the form of a *service function chain*

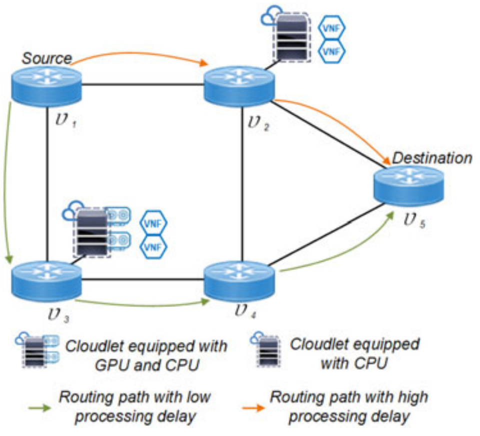

Source
VNF
VNF
v_1
v_2
Destination
v_5
VNF
VNF
v_3
v_4
Cloudlet equipped with GPU and CPU
Cloudlet equipped with CPU
Routing path with low processing delay
Routing path with high processing delay

Fig. 1. An example of the mobile edge cloud.

that consists of a sequence of VNFs, specifying the order of VNFs that its data traffic should pass through. Specifically, denote by $r_k = (s_k, t_k, VNF_k, \rho_k, D_k)$ a user request. Each request r_k requires to transmit data from a source node s_k to a destination node t_k at a packet rate ρ_k within a delay no greater than D_k. Request r_k also requires its traffic to pass through each VNF instance in its service chain VNF_k before reaching destination t_k. We consider that the computing resource demanded by service chain VNF_k is $RC \cdot \rho_k$ [37], [40], where RC is the amount of computing resource to process a packet. For the sake of clarity, we set RC to 1; RC however can be adapted to real environments with different settings of resource consumptions. Without loss of generality, we assume that the total capability of a number of cloudlets is sufficient for the processing of all requests. Thus, the total packet rates of all requests is no greater than δ times of the minimum computing resource capacity of C_{min}, i.e., $\sum_{r_k \in R} \rho_k \le \delta \cdot C_{min}$. However, a single cloudlet may not be enough to process all requests in the system, which means that there are at least two cloudlets for request admissions, i.e., $\delta > 2$. In this paper, we assume that all requests can be admitted by the system. Without loss of generality, we assume that $\delta \le |CL|$. The value of δ thus is in the range of $[2, |CL|]$.

3.3 Soft- and Hard-Affinity

Implementing VNFs in generic CPU cores may degrade its performance. To improve the performance, some packet processing acceleration methods have been proposed [43]. For example, GPU based packet processing acceleration is used to implement IDSes. VNFs that adopt such flow processing acceleration methods favor staying in the cloudlets with GPUs or FPGAs. Otherwise, their performance may degrade significantly. In addition, VNFs usually generate various states in the processing of data traffic, and such states ensure the correct processing of future data packets. Therefore, cloudlet cl_m saves some state information and relevant data of a VNF VNF_k for the future usage of other requests demanding VNF_k too. If we deploy an instance of VNF_k to a cloudlet that is far away from the location of cloudlet cl_m with its state data, the delay of pulling those state data from cl_m might take prohibitively long. The reason is that interconnection fabrics are typically much congested and slower than the high-speed local data storage.

Therefore, an instance of VNF_k prefers to be instantiated in the cloudlets with GPU accelerations or states. This is referred to as *affinity* of VNFs.

In this paper, we consider *soft-affinity* where each VNF_k prefers to be implemented by GPUs of a specified set of cloudlets in the mobile edge cloud; however, it can also be placed into any cloudlet that is not in the specified set and being implemented using CPU resource. Denote by $\mathcal{CL}_k^{af}$ the set of cloudlets that r_k specifies to implement its VNF_k, where $\mathcal{CL}_k^{af} \subset \{CL\}$. An instance of VNF_k can be placed to a cloudlet $cl_m \notin \mathcal{CL}_k^{af}$; however the delay of processing its traffic will be higher as no GPU accelerators can be leveraged and its required states have to be pulled while processing the data traffic. On the other hand, the *hard-affinity* considers that each VNF_k can only be placed to the GPUs of one of its specified cloudlets in $\mathcal{CL}_k^{af}$. The cloudlets in $\mathcal{CL}_k^{af}$ can be obtained in historical information of implementations of similar VNFs. Or, users can select a set of such cloudlets for their initial requests according to the recommendations of network operators, and then update the set for their future requests according to the performance of previous VNF implementations.

3.4 VNF Accelerations in GPUs

We adopt the Pascal architecture of NVIDIA's GPUs. Specifically, each GPU has several graphics processing clusters that are composed by number of streaming multiprocessors. Each streaming multiprocessor has multiple stream processors, with each stream processor executing a single GPU thread. To support the execution VNFs in the GPU architecture, there are several GPU acceleration designs for the implementation of VNFs. We here adopt the design of FlowShader [39] that requires the minimum modification of the software design of each VNF. In FlowShader, each GPU thread is used to process a single data flow and run the whole logic of a VNF. Recall that we consider that each request r_k requires to transmit and process its data traffic at rate of ρ_k. We thus assume that each data flow can be processed by a GPU thread. Specifically, if request ρ_k is assigned to a GPU for processing, there will be a GPU thread assigned to it. This means that the maximum number of requests that can be processed by each GPU is the total number of GPU threads it can run. It must be mentioned that the sizes of data traffic of requests varies, which may cause some GPU cores overloaded. To avoid this, large flows of requests are divided among CPU cores and GPU cores, following the flow balancing method for CPU and GPU of each edge server in FlowShader [39]. The benefit of doing this is to allow any VNF to be accelerated without redesigning the processing logic of its network function. The rationale behind is that by adopting a weak-scaling approach and a flow-level parallelism, similar VNF logic running on CPU is run on each GPU thread. Therefore, the network operator does not need to decide which VNF can or cannot be accelerated by its GPUs, as long as the VNFs are flow-processing network functions and each of its logic can run in a GPU thread.

3.5 End-to-End Delay Requirement

Cloud services usually have delay-sensitive requirements for the processing of their data traffic. We here consider the *end-to-end* delay requirement of a packet processing from the source to destination of each service request, which is a main metric indicating the quality of a VNF instance. In particular, the end-to-end delay requirement D_k of each request r_k specifies the maximum tolerable delay of its packet transfer from the source node s_k to the destination node t_k. It consists of the processing delay of VNF_k, and the transfer delay on each link in the routing path of the data traffic.

For the processing delay of VNF_k in a cloudlet cl_m with a GPU, we consider the flow-parallel acceleration framework in [39], where similar network function logic as that running on CPU can be executed on the CPU with a minimum logic modification. In the framework, the entire flow of a request will be processed by a GPU thread if the request is assigned to a GPU. Therefore, the performance of the parallel flow processing of cloudlet cl_m is limited by the number of GPU threads that can be provided by the GPUs of cl_m. Recall that $Th(cl_m)$ is the number of GPU threads provided by cloudlet cl_m. Following the results in [39], we consider that the processing delay remains a constant if the number of requests that are processed by a GPU is lower than $Th(cl_m)$. Let $d(VNF_k, cl_m)$ be the delay of processing a packet of request r_k in a GPU of cl_m, which is usually a given small constant, if the number of concurrent requests is no greater than $Th(cl_m)$. If VNF_k is implemented in a GPU of cloudlet cl_m, the end-to-end delay of request can be calculated by

$$d(s_k, cl_m) = d(s_k, cl_m) + d(VNF_k, cl_m) + d(cl_m, t_k), \tag{1}$$

where $d(s_k, cl_m)$ is the transmission delay from source s_k to cloudlet cl_m, and $d(cl_m, t_k)$ is the delay from cl_m to destination t_k.

The end-to-end delay of r_k when it is processed by a CPU in cl_m, denoted by $d'(s_k, cl_m)$, thus is

$$d'(s_k, cl_m) = d(s_k, cl_m) + d'(VNF_k, cl_m) + d(cl_m, t_k), \tag{2}$$

where $d'(VNF_k, cl_m)$ is the delay of processing a packet of r_k by the CPU power of cloudlet cl_m.

Let D_k be the maximum delay that is tolerable by request r_k. We then have the end-to-end delay requirement $d(s_k, cl_m) \leq D_k$ and $d'(s_k, cl_m) \leq D_k$.

3.6 Problem Definitions

Given a software-defined mobile edge cloud $G = (V \cup CL, E)$ and a set of user requests R, we first consider the admission of requests in G. We then consider the dynamic admission of requests, by assuming that requests arrive one by one without the knowledge of future arrivals. Specifically, we consider the following two optimization problems.

Problem 1. The soft-affinity-aware throughput maximization problem *in $G(V \cup CL, E)$ is to admit as many requests in R as possible such that the system throughput – the accumulative data volume of admitted requests – is maximized, while VNFs of the admitted requests can be implemented in cloudlets with or without their required hardware accelerators and state information, subject to the constraints of computing resource on each cloudlet cl_m, the capacity constraint of the accelerator of cl_m, the TCAM capacity on each SDN switch $v \in V$, and the end-to-end delay D_k of each request r_k.*

Problem 2. *Consider a dynamic environment where requests arrive one by one without the knowledge of future arrivals.* The online hard-affinity-aware throughput maximization problem *in $G(V \cup CL, E)$ is to find a scheduling that each arrived request is either admitted or rejected immediately such that the accumulative system throughput in a finite time horizon – the accumulative data volume of admitted requests – is maximized, while VNFs are implemented in cloudlets with their required hardware accelerators and state information, subject to the computing resource capacity on each cloudlet cl_m, the TCAM capacity on each $v \in V$, and the end-to-end delay D_k of each request r_k.*

It must be mentioned that the aforementioned optimization problems are NP-hard, which can be shown through a reduction from the generalized assignment problem [28] to a special case of the problems without imposed the end-to-end delay and TCAM constraints, while the former is NP-hard.

3.7 Approximation and Competitive Ratios

The Approximation Ratio. Given a value $0 < \gamma < 1$, a *γ-approximation algorithm* for a maximization problem P_1 is a polynomial time algorithm $\mathcal{A}$ that outputs a solution whose value is no less than γ times the value of an optimal solution for any instance I of P_1, where γ is termed as the approximation ratio of algorithm $\mathcal{A}$.

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

For clarity, the symbols used in this paper are summarized in Table 1.

4 Integer Linear Programming

In this section we formulate the soft-affinity-aware throughput maximization problem as an Integer Linear Program.

Recall that if a cloudlet $cl_m \in \mathcal{CL}_k^{af}$ is selected to admit request r_k, VNF_k may or may not be accelerated by a GPU in cl_m. We thus use an indicator decision variable x_{km} to indicate whether request r_k is assigned to an instance of its VNF in a GPU of cloudlet cl_m. We also use another indicator variable y_{km} to indicate whether request r_k is assigned one VNF instance to cloudlet cl_m for implementation by the CPU resource of cl_m. This means that for a cloudlet cl_m, CPU resource is used to process the traffic of VNF_k instead of a GPU of cl_m if $x_{km} = 0$ and $y_{km} = 1$. The data traffic of r_j then is routed via a shortest path from its source node s_k to the selected cloudlet, and then from the selected cloudlet to its destination t_k. The optimization objective of the soft-affinity-aware throughput maximization problem thus is to

$$\textbf{ILP-SA}: \quad \max \sum_{cl_m \in \mathcal{CL}_k^{af}} \sum_{r_k \in R} x_{km}\rho_k + \sum_{cl_m \in \mathcal{CL}} \sum_{r_k \in R} y_{km}\rho_k, \tag{3}$$

subject to the following constraints.

$$\sum_{cl_m \in \mathcal{CL}} (x_{km} + y_{km}) \leq 1, \qquad \forall r_k \in R \tag{4}$$

$$x_{km} = 0, \quad \forall r_k \in R \text{ and } \forall cl_m \in CL \setminus CL_k^{af} \tag{5}$$

$$x_{km} + y_{km} \leq 1, \quad \forall r_k \in R \text{ and } \forall cl_m \in CL_k^{af} \tag{6}$$

$$\sum_{r_k \in R} (x_{km} + y_{km}) \cdot \rho_k \leq C(cl_m), \quad \forall cl_m \in CL \tag{7}$$

$$\sum_{r_k \in R} x_{km} \leq Th(cl_m), \quad \forall cl_m \in CL_k^{af} \tag{8}$$

$$\sum_{r_k \in R} (x_{km} + y_{km}) \leq N(v), \quad \text{for each } cl_m \text{ and each } v \in p_{s_k, cl_m} \cup p_{cl_m, s_k} \tag{9}$$

$$x_{km} \cdot d(r_k, cl_m) \leq D_k, \quad \forall cl_m, r_k \in R \tag{10}$$

$$y_{km} \cdot d'(r_k, cl_m) \leq D_k, \quad \forall cl_m, r_k \in R \tag{11}$$

$$x_{km}, y_{km} \in \{0, 1\}, \tag{12}$$

where Constraints (4) say that the VNF of each request r_k can only be placed into one cloudlet $cl_m \in \mathcal{CL}$ no matter whether its VNF is implemented in a GPU or not. Constraints (5) indicate that $x_{km} = 0$ for all cloudlets not in $\mathcal{CL}_k^{af}$. This is because the cloudlets not in $\mathcal{CL}_k^{af}$ are not allowed to accelerate the VNF of r_k, by the definition of soft-affinity. Constraints (6) indicate that for a cloudlet with GPUs, the traffic of r_k can be processed by an instance of its VNF_k in a GPU, or an instance of VNF_k in a CPU of cl_m. In other words, y_{km} must be 0 if $x_{km} = 1$, meaning that r_k is accelerated by a GPU of cl_m. Also, x_{km} has to be 0 if $y_{km} = 1$, indicating that the request is implemented via a CPU in cl_m. However, x_{km} and y_{km} can be both zeros, saying that other cloudlets will implement r_k. Constraints (7) show that the computing resource demanded by the requests assigned to cl_m should not exceed its capacity $C(cl_m)$ no matter whether its traffic is implemented in a GPU. The rationale behind is that, even if the traffic of a request is processed by a GPU, CPU capacity is still needed to process partial of the traffic of the request [39]. Also note that we adopt the popular proportional resource consumption model for each user request, following many existing studies [1], [4], [46]. This means that the resource consumption of a request r_k is proportional data rate ρ_k. It is true that CPU resource consumption of processing a unit of data rate varies for different types of VNFs. Constraints (7) can be extended easily to consider such scenarios, by adding a scaling factor for each request r_k in the RHS of Constraints (7). Constraints (8) ensure that the number of requests processed by the GPUs of a cloudlet cannot exceed the number of GPU threads in cl_m, i.e., $Th(cl_m)$, such that the acceleration performance of packet processing is guaranteed. Constraints (9) make sure that the forwarding table size of each switch that forwards the pre-processing traffic or post-processing traffic of r_k should not be violated. Note that we assume that only one forwarding rule in a switch is enough for both pre- and post-processing traffic. Constraints (10) and (11) say that the end-to-end delay requirement D_k of each admitted request r_k should not be violated. Constraints (12) impose the integral constraint of the indicator variables x_{km} and y_{km}.

It must be mentioned that the ILP solution may take prohibitively long to deliver an exact solution for the problem when its size is large, i.e., with hundreds of nodes in the

TABLE 1
Symbols

Symbols	Meaning
$G = (V \cup CL, E)$	a mobile edge network deployed in a wireless metropolitan area with a set V of switches and a set CL of cloudlets.
v and $N(v)$	a switch in V and the number of forwarding entries in the TCAM of switch $v \in V$.
cl_m	a cloudlet in CL.
$C(cl_m)$ and C_{min}	the capacity of computing resource of cl_m and the minimum computing capacity.
$Th(cl_m)$	the number of GPU threads of each cloudlet $cl_m \in CL$.
RC	the amount of computing resource to process a packet.
e and d_e	a link in E and the delay of transmitting a unit packet data along link e.
$p_{v,v'}$	the shortest path between $v \in V$ and $v' \in V$.
$d_{v,v'}$	the delay for transmitting a unit of data packets along the shortest path between $v \in V$ and $v' \in V$.
r_k, s_k, t_k	a user request and its source, destination.
ρ_k	the volume of data packets that r_k need to transmit.
VNF_k and D_k	the VNF of r_k and the delay requirement of r_k.
CL_k^{af}	the set of locations with the GPUs that r_k requires its data to be processed.
$d(VNF_k, cl_m)$	the delay for processing the packets of r_k in a GPU of cloudlet cl_m.
$d'(VNF_k, cl_m)$	the delay for processing the packets of r_k in a CPU of cloudlet cl_m.
d_{r_k,cl_m}	the end-to-end delay for transmitting the packets of r_k from s_k to t_k.
$d(s_k, cl_m)$	the end-to-end delay of request r_k if its implemented in the GPU of cl_m.
$d'(s_k, cl_m)$	the end-to-end delay of request r_k if its implemented in the CPU of cl_m.
x_{km}	an indicator variable that indicates whether r_k is deployed at the GPU of cl_m.
y_{km}	a binary decision variable that shows whether r_k is assigned to the CPU of cl_m.
R^{ha} and R^{sa}	the set of requests with hard-affinity and soft-affinity requirements.
x_{km}^*	the optimal fractional solution to the proposed **ILP-SA**.
α	a constant value that illustrates the relationship between number of cloudlets $\lvert CL\rvert$ and capacity of TCAM in each SDN-enabled switch v.
δ	a constant value that illustrates the total packet rates of all requests is smaller than δ times of the minimum computing resource capacity of C_{min}.
P_m and I^*	the shadow price of each cloudlet cl_m and the maximum resource usage by a request with a unit packet rate.
Δ	a constant value that guides resource reservations for future request admissions.
ϵ	a constant value that serves as a tradeoff between the competitiveness of the proposed online algorithm and the degree of resource violations.
γ_k, η_m, ζ_m, κ_{mv}, and ω_{mk}	the dual variables for Constraints (14), (15), (16), (17), and (18), respectively.
X	an independent random variable that represents the amount of request traffic that is assigned to VNFs in cl_m for processing.
$E(X)$ and $Var(X)$	the expectation and variance of the event X.
$Pr[X \geq C(cl_m)]$	the probability of violating the computing capacity of cloudlet cl_m.
Y	an independent random variable that represents the number of requests whose traffics are forwarded by switch $v \in V$.
V_k^{pre}	the set of the switches that are used to forward the pre-processed traffic of r_k.
$Pr[v \in V_k^{pre}]$	the probability of using switch v to forward the pre-processed traffic of r_k.
$Pr[Y \geq N(v)]$	the probability of violating the forwarding table size of switch $v \in V$.
W	the accumulative packet rate due to the randomized algorithm `Appro`.
β	the approximation of the proposed algorithm `Appro`.
x'	the feasible solution to the problem calculated by the randomized algorithm `Appro`.
$L_m(k)$	the total packet rate of admitted requests after the admission of request r_k.
$\eta_m(k)$	the value of dual variable η_m after the admission of request r_k.

network. Even if the exact solution is eventually obtained, it may no longer be valuable, and the result cannot be used for improving the network performance due to dynamic evolution of resources in the network.

5 Algorithms for the Soft-Affinity-Aware Throughput Maximization Problem

In this section, we propose a heuristic for the soft-affinity-aware throughput maximization problem. We first consider a special case of the problem where a request can only be assigned to a cloudlet in its specified set of cloudlets and accelerated by a GPU thread. We refer to this special problem as *the hard-affinity-aware throughput maximization problem*, and we propose an exact and approximate solutions to it, respectively. We then devise an efficient heuristic for the soft-affinity-aware throughput maximization problem, based on the proposed approximate solution.

5.1 An Exact Solution to the Hard-Affinity-Aware Throughput Maximization Problem

In the hard-affinity-aware throughput maximization problem, a request r_k has to be admitted into a cloudlet $cl_m \in CL_k^{af}$ and its VNF_k is implemented by a GPU of cl_m. We re-use the indicator variable x_{km} in **ILP-SA** to indicate whether request r_k is assigned to an instance of its VNFs in cloudlet $cl_m \in CL_k^{af}$ to process its traffic. The optimization objective of the hard-affinity throughput maximization problem is

$$\textbf{ILP-HA}: \quad \max \sum_{cl_m \in CL_k^{af}} \sum_{r_k \in R} x_{km} \rho_k, \tag{13}$$

subject to the following constraints.

$$\sum_{cl_m \in CL_k^{af}} x_{km} \leq 1, \qquad \forall r_k \in R \tag{14}$$

$$\sum_{r_k \in R} x_{km} \cdot \rho_k \leq C(cl_m), \qquad \forall cl_m \in CL \tag{15}$$

$$\sum_{r_k \in R} x_{km} \leq Th(cl_m), \qquad \forall cl_m \in CL \tag{16}$$

$$\sum_{r_k \in R} x_{km} \leq N(v), \quad \text{for each } cl_m \text{ and each } v \in p_{s_k, cl_m} \cup p_{cl_m, s_k} \tag{17}$$

$$x_{km} \cdot d(r_k, cl_m) \leq D_k, \quad \forall cl_m, r_k \in R \tag{18}$$

$$x_{km} \in \{0, 1\}, \tag{19}$$

where Constraints (14) indicate that the VNF of each request r_k can only be placed into one cloudlet $cl_m \in \mathcal{CL}_k^{af}$. Constraints (15) say that the CPU resource which will be used to requests that are assigned to each cloudlet cl_m should not exceed its capacity $C(cl_m)$. Constraints (16) make sure the total number of requests that are assigned to the GPUs of cl_m is no greater than the number $Th(cl_m)$ of GPU threads in cl_m. Constraints (17) guarantee that for each switch that forwards the pre-processing traffic or post-processing traffic of r_k will not exceed its forwarding table size $N(v)$. Constraints (18) say that the end-to-end delay requirement D_k of each admitted request r_k should not be violated. Constraints (19) impose the integral constraint on the indicator variable x_{km}.

5.2 An Approximation Algorithm for the Hard-Affinity-Aware Throughput Maximization Problem

Due to the high computational complexity of the proposed ILP solution, in the following we propose an efficient approximation algorithm with a good approximation ratio for the problem.

The basic idea of the proposed approximation algorithm is to adopt the randomized rounding technique [29] on the ILP solution. Specifically, we first relax the proposed **ILP-HA** by assuming x_{km} is a real value in the range of $[0, 1]$. A fractional solution to the constrained maximization problem is then obtained in polynomial time. Let $\boldsymbol{x}^*$ be the optimal fractional solution. Let OPT be the optimal solution to the hard-affinity-aware throughput maximization problem. The value of $\boldsymbol{x}^*$ is an upper bound of the value of the optimal solution OPT. We then round the fractional x_{km} into $0/1$ with probability x^*_{km}.

We now describe the randomized algorithm. We first relax the **ILP-HA** to a Linear Program (LP) by relaxing Constraints (19) into

$$0 \leq x_{km} \leq 1, \tag{20}$$

subject to Constraints (14), (15), (16), (17), (18), and (20).

The obtained linear program is referred to as **LP-HA**. A fractional solution $\boldsymbol{x}^*$ is obtained by solving the **LP-HA**.

We then round the fractional solution to an integral solution $\boldsymbol{x}$ to the original problem. We round x_{km} to 1 with probability x^*_{km} if $x^*_{km} \geq \frac{\delta^2}{1+\delta^2}$, where $\frac{\delta^2}{1+\delta^2}$ illustrates a threshold that ensures the possibility of capacity violence on a cloudlet $cl_m \in CL$ is acceptable. We thus consider $Pr[x_{km} = 1] = x^*_{km}$, which also means that we round x_{km} to 0 with probability $1 - x^*_{km}$, if $x^*_{km} \geq \frac{\delta^2}{1+\delta^2}$ or $x^*_{km} \leq \frac{\delta^2}{1+\delta^2}$. Note that $Pr[x_{km} = 1] = x^*_{km}$ represents the probability that the VNF_k of request r_k is assigned to cloudlet cl_m. The intuition of setting a threshold, i.e., $\frac{\delta^2}{1+\delta^2}$, on the value of x^*_{km} is to filter out the impact of some trivial possible assignments with low probability. It must be mentioned the performance of the proposed approximation algorithm largely depends on the value of δ. For example, when δ is small, the algorithm will have a smaller value for threshold $\frac{\delta^2}{1+\delta^2}$, implies that the algorithm is loose in selecting solutions for request admissions. On the other hand, when the algorithm prefers to select good solutions with high probability, it may set a larger value for θ.

The detailed algorithm is given in Algorithm 1.

Algorithm 1. `Appro`

```
Input: $G = (V \cup CL, E)$, a set of requests $R$.
Output: The assignment of the each request $r_k \in R$ to a cloudlet.
1: Solve the relaxed version of ILP-HA, i.e., LP-HA;
2: for each $r_k \in R$ do
3:     Implement $VNF_k$ of each request $r_k \in R$ in $cl_m$ with a
       probability of $x^*_{km}$, if $x^*_{km} \geq \frac{\delta^2}{1+\delta^2}$;
```

5.3 A Heuristic for the Soft-Affinity-Aware Throughput Maximization Problem

We now propose a heuristic for the soft-affinity-aware throughput maximization problem, based on algorithm `Appro` for its special case. Recall that the problem objective is to maximize the accumulative data volume of admitted requests. To this end, a request may be assigned to a cloudlet by using the CPU resource of the cloudlet to process its traffic if its delay requirement can be met, because CPU capacities usually can process requests with large data volumes of requests. The basic idea of the proposed algorithm thus is to assign a request r_k to a cloudlet in $CL \setminus CL_k^{af}$ as long as its delay requirement can be met; otherwise, it will be preferably assigned to a GPU of a cloudlet in CL_k^{af}.

We first rank requests in R into non-increasing order of their data volume ρ_k. We then partition the ranked list into two sublists by going over the ranked list one by one. Let R^{ha} be the set of requests that have to be assigned GPUs of cloudlets, i.e., the ones with hard-affinity requirements. The set with the rest requests is denoted by R^{sa}. Specifically, we add a request r_k to R^{ha} as long as its delay requirement cannot be met if it is implemented using the CPU capacity of a cloudlet; otherwise, it is added into R^{sa}. We then invoke algorithm `Appro` to assign requests in R^{ha} to cloudlets. We admit requests in R^{sa} one by one until the CPU capacity of each cloudlet is saturated. The detailed steps are shown in Algorithm 2, which is referred to as algorithm `Heu`.

5.4 Performance Analysis

In the following, we analyze the performance of algorithm `Appro`.

Lemma 1. *For the solution obtained from Algorithm 1, we claim (1) the probability of violating the computing resource capacity on a cloudlet is no more than $\frac{1}{(\delta-1)^2}$ with $\delta > 2$; (2) the probability of violating the GPU thread limitation of each cloudlet cl_m is no more than $\frac{1}{(\delta-1)^2}$; and (3) the probability of violating the forwarding table size of each switch $v \in V$ is no more than $\frac{1}{|CL|(\alpha-1)^2}$, where α is a given integer constant with $\alpha > 1$.*

Please see the appendix for the detailed proof, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TC.2020.3041629.

Theorem 1. *Given a mobile edge cloud $G = (V \cup CL, E)$ with a set of cloudlets in CL, a set of requests R with each request r_k specifying its VNF being implemented in a set CL_k^{af} of cloudlets, the CPU computing and GPU resource constraints of each cloudlet $cl_m \in CL$, and the forwarding table size limitation of each switch $v \in V$, there is an approximation algorithm,* `Appro`*, for the hard-affinity-aware throughput maximization problem in G, which can achieve an approximation ratio of $\frac{(1+\delta)^2\cdot(\delta-2)}{\delta\cdot(\delta-1)^2}$ with high probability, where δ is a given constant with $2 \leq \delta \leq |\mathcal{CL}|$.*

Please see the appendix for the detailed proof, available in the online supplemental material.

Algorithm 2. `Heu`

```
Input: $G = (V \cup CL, E)$, a set of requests $R$.
Output: The assignment of the each request $r_k \in R$ to a cloudlet.
1: Rank the requests in $R$ into non-increasing order in terms of
   their data volume $\rho_k$;
2: for each request $r_k$ in the ranked list do
3:     Add $r_k$ to $R^{ha}$ and remove it from the ranked list, if its
       delay requirement $D_k$ cannot be met if it is implemented
       by the CPU capacity of a cloudlet;
4: Invoke algorithm Appro to admit requests in $R^{ha}$;
5: Add all the rest requests in the ranked list to $R^{sa}$, and admit
   the requests in $R^{sa}$ one by one greedily;
```

6 Algorithm for the Online Hard-Affinity-Aware Throughput Maximization Problem

In this section, we propose an online algorithm with a good competitive ratio for the online hard-affinity-aware throughput maximization problem.

6.1 Algorithm Overview

The basic idea of the proposed online algorithm is to adopt the primal-dual-updating approach [3] for the problem, which maintains *shadow price* variables for cloudlets and switches in the software-defined mobile edge cloud G. As we assume that requests arrive one by one, myopic requests admissions may harm the admissions of future requests, thereby reducing the accumulative system throughput. We here propose an efficient *admission control policy* that guides requests admissions. Specifically, on the arrival of each request r_k, the online algorithm compares the price of the cheapest cloudlet in terms of the shadow price to a specially calibrated threshold, to determine whether r_k should be admitted or not. For clarity, the dual of the **ILP-HA** is given as follows.

Let γ_k, η_m, ζ_m, κ_{mv}, and ω_{mk} be the dual variables for Constraints (14), (15), (16), (17), and (18), respectively. Since our objective is to maximize the accumulative packet rate of admitted requests in R. Its duel objective is to minimize the cost, i.e.,

$$\min \gamma_k + C(cl_m)\cdot\eta_m + Th(cl_m)\cdot\zeta_m + N(v)\cdot\kappa_{mv} + D_k\cdot\omega_{mk}, \tag{21}$$

subject to,

$$\sum_{r_k\in R}\gamma_k + \sum_{cl_m\in CL}\rho_k\eta_m + \sum_{cl_m\in CL}\zeta_m + \sum_{cl_m\in CL}\sum_{v\in CL_k^{af}}\kappa_{mv} + \sum_{cl_m\in CL}\sum_{r_k\in R} d(r_k, cl_m)\omega_{mk} \geq \rho_k, \tag{22}$$

which can be re-written as

$$\sum_{r_k\in R}\gamma_k \geq \rho_k - \sum_{cl_m\in CL}\rho_k\eta_m - \sum_{cl_m\in CL}\zeta_m - \sum_{cl_m\in CL}\sum_{v\in CL_k^{af}}\kappa_{mv} - \sum_{cl_m\in CL}\sum_{r_k\in R} d(r_k, cl_m)\omega_{mk}. \tag{23}$$

6.2 Online Algorithm

We first define the shadow price P_m of each cloudlet cl_m, the maximum resource usage I^* of requests in R, and a constant Δ to guide resource reservations for future requests. Specifically, the shadow price P_m represents the marginal increase of strengthening the resource capacity constraint on each cloudlet, the GPU thread limitation constraint of each cloudlet, and the forwarding table size constraint on each switch, which is defined as

$$P_m = \left(\frac{\kappa_{mv}}{\rho_k} + \eta_m + \frac{\zeta_m}{\rho_k}\right). \tag{24}$$

Then, the maximum resource usage I^* by a request with a unit packet rate can be defined as

$$I^* = \max\left\{1, \max_{cl_m\in CL, r_k\in R}\left(\frac{d(r_k, cl_m)}{\rho_k}, \frac{1}{N(v)\rho_k}, \frac{1}{\rho_k}\right)\right\}. \tag{25}$$

We now provide a flexible way of tuning resource reservations for future requests. Specifically, we introduce a parameter ϵ with $0 < \epsilon \leq 1$ and

$$\Delta = I^*/\epsilon. \tag{26}$$

This serves as a tradeoff between the competitiveness of the proposed algorithm and the degree of resource violations. In general, a smaller value of ϵ means a conservative way of reserving resources for future requests. It thus avoids severe resource violations; however, it may miss the opportunity of admitting future requests with higher demands and thus reduce the system throughput. This feature will be introduced in the performance analysis of the algorithm later. The rationale of the proposed admission policy is to use a pre-defined threshold of the resource usage of r_k to decide whether the request should be admitted. Specifically, if r_k's resource usage in the cloudlet with the minimum shadow price is higher than the given threshold, the request will be rejected. Namely, request r_k will be rejected if

$$P_{m^*} \geq 1 - \frac{(d(r_k, cl_m))^2}{\Delta \cdot D_k \cdot \rho_k}, \tag{27}$$

where $P_{m^*} = \min_{cl_m \in CL}\{P_m\}$.

Given the admission control policy, the dual variables will be updated when request r_k is admitted. We now describe the update rules for the dual variables of the dual program. Each dual variable is initially set to zero, and then updated according to the following rules:

$$\gamma_k \leftarrow \rho_k(1 - P_{m^*}), \tag{28}$$

$$\eta_{m^*} \leftarrow \eta_{m^*}\Big(1 + \frac{\rho_k}{C(cl_{m^*})}\Big) + \frac{\rho_k}{\Delta \cdot C(cl_{m^*})}, \tag{29}$$

$$\zeta_{m^*} \leftarrow \zeta_{m^*}\Big(1 + \frac{1}{Th(cl_{m^*})}\Big) + \frac{1}{\Delta \cdot Th(cl_{m^*})}, \tag{30}$$

$$\kappa_{m^*v} \leftarrow \kappa_{m^*v}\Big(1 + \frac{1}{N(v)}\Big) + \frac{1}{\Delta \cdot N(v)}, \tag{31}$$

$$\omega_{m^*k} \leftarrow \frac{d(r_k, cl_{m^*})}{\Delta \cdot D_k}. \tag{32}$$

The detailed algorithm is given in Algorithm 3, which is also referred to as Algorithm `Online`.

Algorithm 3. `Online`

```
Input: $G = (V \cup CL, E)$, a set of requests $R$ that arrive into the
    system one by one without the knowledge of their future arrivals.
Output: The assignment of the each request $r_k \in R$ to a cloudlet.
1: for each request $r_k$ do
2:     Let $CL_k$ be the set of data centers that can meet the
       end-to-end delay requirement of $r_k$;
3:     Find the cloudlet in $CL_k$ that has the minimum shadow
       price, i.e, $P_{m^*}$;
4:     if $P_{m^*} \geq 1 - \frac{(d(r_k,cl_{m^*}))^2}{\Delta \cdot D_k \cdot \rho_k}$ then
5:         Reject request $r_k$;
6:     else
7:         Implement request $r_k$ in cloudlet $cl_{m^*}$;
8:         Update the dual variables following rules (28), (29), (31),
           and (32);
```

6.3 Algorithm Analysis

The rest is to show the dual feasibility of the dual variables in the dual program, and then analyze the competitive ratio of the proposed online algorithm.

Lemma 2. *The updating rules for dual variables (28), (29), (31), and (32) always meet the dual feasibility requirement of the dual program.*

The detailed proof is shown in the appendix of this paper, available in the online supplemental material.

Lemma 3. *Whenever a request r_k is admitted, the increase of the objective of the dual program is no more than $(1+\epsilon)\rho_k$, where ϵ is a parameter with $0 < \epsilon \leq 1$.*

The detailed proof of this lemma is shown in the appendix of this paper, available in the online supplemental material.

We proceed by analyzing the upper bound on the resource violation ratios of each constraint of the **ILP** in the following lemma.

Lemma 4. *The delay requirement of each admitted request is met. The violation of the computing capacity constant, the GPU thread limitation constraint, and the forwarding table size constraint is at most by a factor of $O(\log N + \log(1/\epsilon))$, where $N = \max\{C(cl_m), N(v), D_k\}$.*

Please see the appendix for the detailed proof, available in the online supplemental material.

We finally analyze the competitive ratio of the proposed Algorithm 3.

Theorem 2. *Given a mobile edge cloud $G = (V \cup CL, E)$ with a set of cloudlets in CL, a set of requests R that arrive into the cloud one by one without the knowledge of future arrivals, the set CL_k^{af} of cloudlets that each request r_k specifies to implement its VNF, the CPU computing and GPU resource constraints of each cloudlet $cl_m \in CL$, and the forwarding table size limitation of each switch $v \in V$, there is an online algorithm,* `Online`*, which has a competitive ratio of $(1-3\epsilon)$, where ϵ is a constant with $0 < \epsilon < 1/3$.*

Please see the appendix for the detailed proof, available in the online supplemental material.

7 Performance Evaluation

In this section, we evaluate the performance of the proposed algorithms by simulations and implementations in the real test-bed.

7.1 Parameter Settings

We consider a mobile edge cloud by varying network size from 50 to 250, where the number of cloudlets in the network is set to 10 percent of the network size, and the cloudlets are randomly co-located with switches. We also utilize a real network AS1755 [31]. We consider that CPU computing capacity of a cloudlet varies from [1,500, 2,500] amount of packet rates, where the computing resource of processing a unit packet rate RC is randomly drawn from [4, 12] MHz [23]. Each switch can install $2 \cdot |CL|$ forwarding rules. Six different types of GPU-accelerated network functions: Flow Monitor (FM), Firewall (FW), Load Balance (LB), IPv4 Router, Network Intrusion Detection System (NIDS) and Internet Protocol Security (IPSec) are considered, and the delay a VNF instance processes a unit packet rate is randomly generated from the range [0.5, 1.2] ms. We consider two types of service chains: FM-FW-LB [42] and IPv4 Router-NIDS-IPSec [45]. The packet rate of each request is randomly drawn from [50, 200] per second. The delay requirement of each request is randomly drawn from [0.1, 1] seconds. We consider two types of GPU, NVIDIA Titan X Pascal and NVIDIA Titan Xp. NVIDIA Titan X Pascal has 6 graphics processing clusters and 30 SMs, and each consists of 128 SPs, resulting in 3584 SPs in total [42] and NVIDIA Titan Xp has 3840 SPs respectively [45]. For each GPU-accelerated VNF instance, we assume that GPU can reduce the

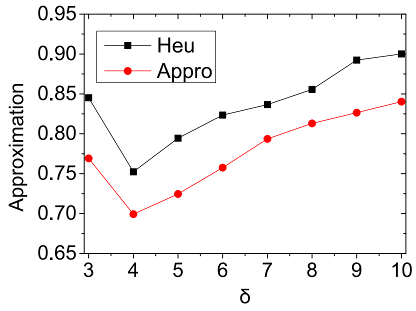


Fig. 2. Approximation ratio of algorithm Heu and Appro against LP with different δ.

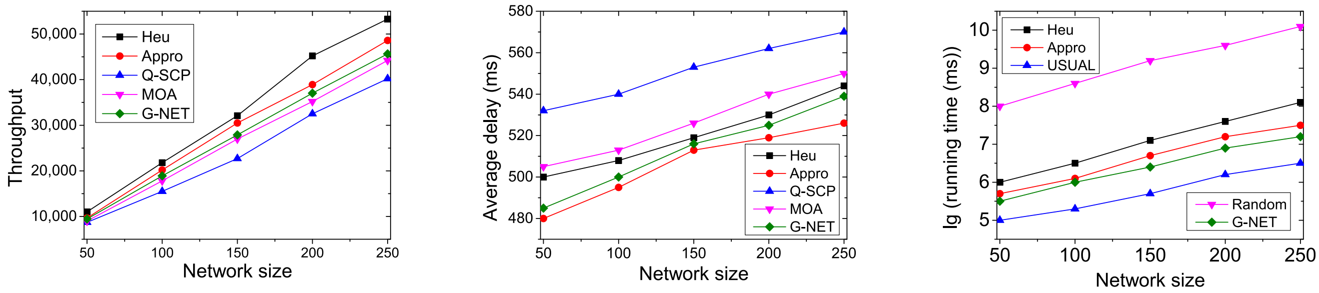
process delay by 25 percent [42]. The running time of each algorithm is obtained based on a machine with a 3.70 GHz Intel i7 Hexa-core CPU and 16 GiB RAM.

We evaluate the performance of the proposed algorithms against the following benchmarks. The first one is a modified version of the solution in [43], which is referred to as G-NET. In algorithm G-NET, a GPU-level scheduling of VNFs is adopted. Specifically, G-NET adopts a greedy approach that assigns a request with a minimum data rate to a GPU that can meet its demand and delay requirements greedily. We however deal with placing VNFs to cloudlets that are different from the study in [43]. To make the comparison fair, we only compare the greedy assignment approach in [43]. Specifically, the greedy approach greedily assigns a request with minimum packet rate to a cloudlet that can satisfy the delay requirement of r_k and implement VNF instances of r_k at the cloudlet. The second benchmark we consider is the one in [46], which is referred to as Q-SCP. For each request r_k, Q-SCP finds a shortest path and selects a candidate cloudlet on the path meeting delay requirement of r_k. The third benchmark we adopt is a multi-objective-based optimization algorithm in [47], which is referred to as MOA. It formulated the problem as a multi-objective integer Linear Programming and then transformed the problem to a single-objective optimization problem to solve.

7.2 Performance Evaluation by Simulations

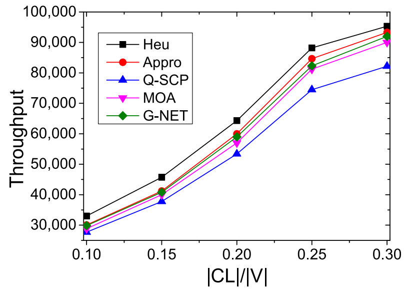
We first compute the approximation ratio of algorithm Heu and Appro by setting the network size to 50, fixing the ratio $|CL|/|V|$ at 0.2 and varying the value of δ from 2 to $|CL|$. Recall that obtaining the optimal solution to the **ILP** may take prohibitively long for a large problem size. To obtain the approximation ratio of algorithm Heu and Appro, we estimate the optimal solution by relaxing the integer decision variables of the **ILP** into real values. The obtained fractional solution is an upper bound of the optimal solution. It must be mentioned that this estimation on the optimal solution is very conservative, the real approximation ratio will be much better than the ratio calculated based on the estimated optimal solution. The results are shown in Fig. 2, from which we can see that with the increase on the value of δ, the approximation ratio first deteriorates when $2 < \delta \leq 4$ and then keeps increasing. This is because the system throughput of the proposed algorithm is impacted by two main reasons: the volume of the data packets of requests in the system and the value of the threshold in algorithms Heu and Appro, i.e., $\frac{\delta^2}{1+\delta^2}$. When $2 \leq \delta \leq 4$, with the increase on δ, there are more requests in the system, leading to the deterioration of the approximation ratio. However, when $\delta \geq 4$, algorithms Heu and Appro behave more like **ILP** because $\frac{\delta^2}{1+\delta^2}$ approaches 1, which is large enough to approach a near optimal solution. We can also see from Fig. 2 that the approximation ratio of algorithm Appro in the worst case is no less than 0.7, when $\delta = 4$, which is consistent with the theoretical results in Theorem 1. When δ approaches $|CL|$, the approximation ratio of algorithms Heu and Appro are around 0.90 and 0.84, respectively, which indicates that the performance of algorithms Heu and Appro are very promising.

We then evaluate the performance of algorithms Heu and Appro against algorithms G-NET, Q-SCP and MOA in terms of the system throughput, average delay and running time, by varying the network size from 50 to 250, while fixing the ratio $|CL|/|V|$ at 0.1. From Fig. 3a, we can see that the system throughput of algorithm Heu is 10 percent higher than that of algorithm Appro. However, when comparing the average delay, the performance of algorithm Heu is worsen than that of algorithm Appro, as depicted in Fig. 3b. This is because algorithm Heu sacrifices the QoS of user requests to have a higher throughput. Besides, we can see that that the throughput achieved by algorithm Appro is consistently higher than that of algorithms G-NET and Q-SCP, respectively, since algorithms G-NET and Q-SCP do not consider global information, leading to lower utilization of computing resources in the system. Although algorithms MOA and Appro have the similar performance behaviors, algorithm MOA takes a longer running time, as illustrated in Fig. 3c. We can see from Fig. 3b that algorithms Heu, Appro and MOA have lower delays than algorithm Q-SCP, because some

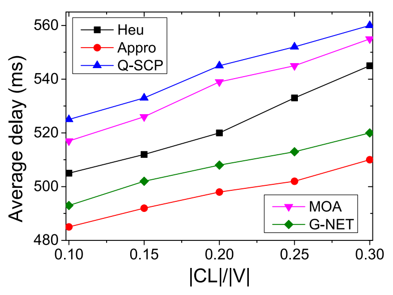


(a) System throughput. (b) Average delay. (c) Running time.

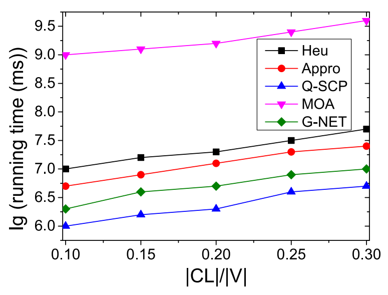
Fig. 3. The performance of algorithms Appro, Heu, G-NET, Q-SCP, and MOA.



(a) System throughput.

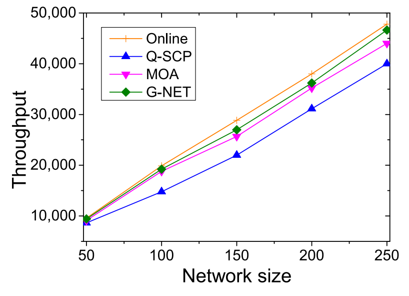


(b) Average delay.

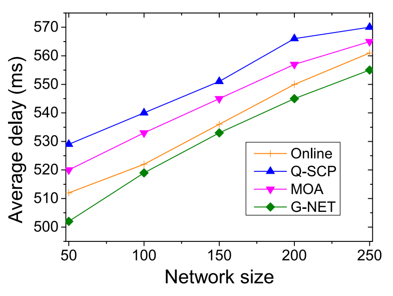


(c) Running time.

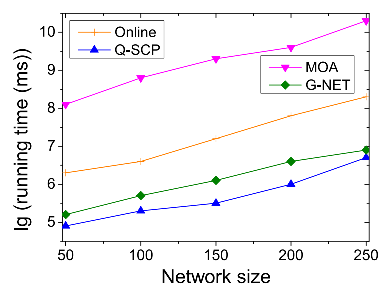
Fig. 4. The performance of algorithms `Appro`, `Heu`, `G-NET`, `Q-SCP`, and `MOA` in the real network AS1755.



(a) System throughput.

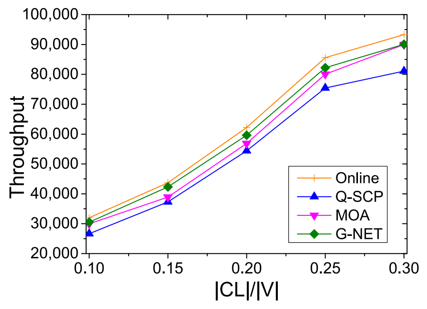


(b) Average delay.

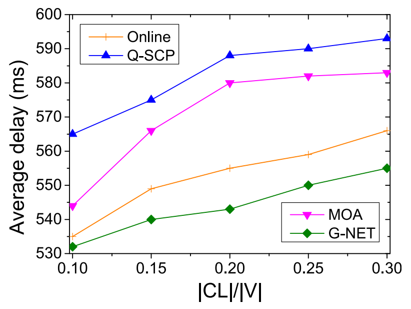


(c) Running time.

Fig. 5. The performance of algorithms `Online`, `G-NET`, `Q-SCP`, and `MOA`.



(a) System throughput.



(b) Average delay.

Fig. 6. The performance of algorithms `Online`, `G-NET`, `Q-SCP`, and `MOA` in the real network AS1755.

requests may be assigned to cloudlets without hardware accelerators, making the processing delay very high and leading to the requests being rejected due to the delay requirement violations.

We also evaluate the performance of algorithms `Heu`, `Appro`, `G-NET`, `Q-SCP` and `MOA` in the real network AS1755, by varying the ratio $|CL|/|V|$ from 0.1 to 0.3. Similarly, we can see from Figs. 4a and 4b that algorithm `Heu` has a higher throughput than algorithm `Appro` but algorithm `Appro` can provide a lower latency services for mobile users. We can also see from Fig. 4a that the throughput is increasing with the growth of ratio $|CL|/|V|$. The reason is that having the network size fixed, the number of cloudlets increases with the growth of the ratio $|CL|/|V|$, which means more computing resource can be used to admit more requests. We can also see that algorithm `Appro` has higher throughput than algorithms `G-NET`, `Q-SCP` and `MOA`, respectively. Similar results on delays as Fig. 3b are shown in Fig. 4b. From Fig. 4c, we can see that algorithm `MOA` has the worst running time among the three comparison algorithms.

We now investigate the performance of algorithm `Online` against algorithms `G-NET`, `Q-SCP` and `MOA` in terms of the system throughput, average delay and running time, by varying the network size from 50 to 250 while fixing the ratio $|CL|/|V|$ at 0.1. We can see from Fig. 5a that algorithm `Online` has a higher system throughput than that of algorithm `USUAL`. From Fig. 5b, we can see that the average delay increases with the growth of network size. This is because in larger networks the source and destination of a request may be farther than that in smaller networks. Also, the performance gap between algorithms `Online` and `Q-SCP` is enlarging with the growth of the network size. The reason is that there are more cloudlets in larger networks; considering that each request has a fixed number of locations that can meet its data locality and hardware affinity requirements, the impact of data locality and hardware affinity requirements is enlarged. Similarly, algorithms `Online` and `MOA` have similar performance, but the latter has a longer running time.

We finally investigate the performance of algorithms `Online`, `G-NET`, `Q-SCP` and `MOA` in the real network

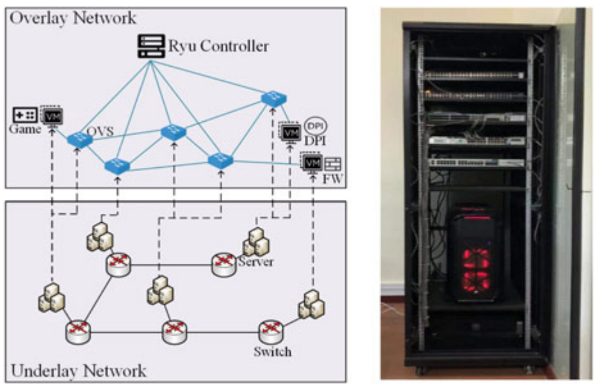


Fig. 7. Test-bed.

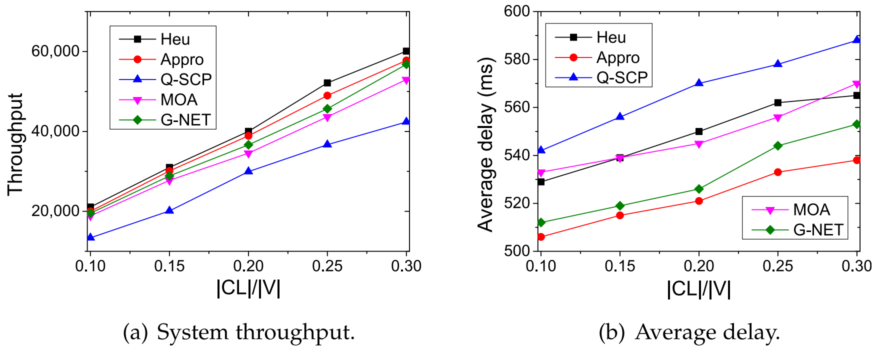
AS1755 [12], by varying the ratio $|CL|/|V|$ from 0.1 to 0.3. The results are shown in Fig. 6. From Fig. 6a, it can be seen that the system throughput increases with the growth of the ratio $|CL|/|V|$ from 0.1 to 0.2. It then keeps stable when the ratio $|CL|/|V|$ increases from 0.2 to 0.3. The reason is that a higher $|CL|/|V|$ that it needs means more cloudlets to implement more requests. However, the throughput becomes stabilized when $|CL|/|V| \geq 0.2$ for the reason that most requests are rejected because of the delay requirement violation. The average delay by algorithm `Online` is lower than algorithms `Q-SCP` and `MOA`. We can see from Fig. 6b that algorithm `G-NET` has a lower delay than that of algorithm `Online`, this is because it first considers the GPU-enabled cloudlet for service chain placement, which leads to the decrease on its system throughput.

7.3 Performance Evaluation in a Real Test-Bed

We implement the proposed algorithms `Appro`, `Heu` and `Online` in a real test-bed with five h3c-S5560X-30C-EI switches and each switch is attached with a server with Intel 8-core i7-4720 CPU. An overlay network following AS1755 topology is constructed based on the physical network of the test-bed using VxLan [35], as shown in Fig. 7. A Ryu controller is used to control the overlay network, and the proposed online algorithms are implemented as a Ryu component. We use Openflow 1.3 [25] to transfer control messages between controller and switches. All the rest parameter settings are the same as the ones in simulations.

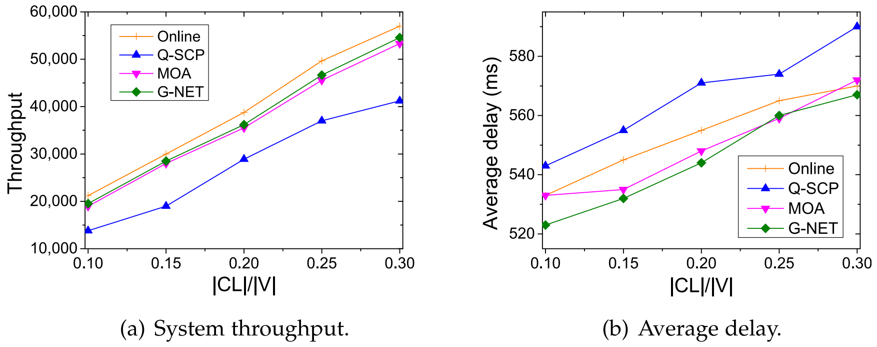
We first evaluate the performance of algorithms `Heu`, `Appro` against algorithms `G-NET`, `Q-SCP` and `MOA` in the test-bed, by varying the ratio $|CL|/|V|$ from 0.1 to 0.3. Results are shown in Fig. 8. From Fig. 8a, we can see that algorithms `Heu` and `Appro` achieve higher throughput than that of algorithms `G-NET`, `Q-SCP` and `MOA`. However, we can see from Fig. 8b that algorithm `MOA` has a longer delay than that of algorithm `Heu` when $|CL|/|V|$ is 0.2. It must be mentioned that the objective of algorithm `Heu` is maximizing the system throughput instead of minimizing the delay; the delay requirements of requests of both algorithms `Heu` and `MOA` are all met. This certifies the proposed online algorithm can deliver a higher throughput than its counterparts in real environments.

We then evaluate the performance of algorithms `Online` against algorithms `G-NET`, `Q-SCP` and `MOA` in the test-bed, by varying the ratio $|CL|/|V|$ from 0.1 to 0.3. Results are shown in Fig. 9. From Fig. 9a, we can see that algorithm `Online` delivers higher system throughput than algorithms



(a) System throughput. (b) Average delay.

Fig. 8. The performance of algorithms `Heu`, `Appro`, `G-NET`, `Q-SCP`, and `MOA` in the test-bed.



(a) System throughput. (b) Average delay.

Fig. 9. The performance of algorithms `Online`, `G-NET`, `Q-SCP`, and `MOA` in the test-bed.

G-NET and MOA when $|CL|/|V|$ and algorithm Q-SCP has the lowest throughput as expected. This certifies the proposed online algorithm can deliver a higher throughput than its counterparts in real environments. From Fig. 9b, we can see that the average delay of algorithm Online is only slightly higher than that of algorithm MOA, but achieves a higher throughput instead. Similarly, algorithm Online considers the global information and achieves higher throughput while algorithms G-NET and MOA focus more on delay of service chain placement.

8 Conclusion and Future Work

In this paper, we considered the locality- and hardware affinity-aware VNF placement in a mobile edge cloud by formulating two optimization problems. We provided an ILP solution for the soft-affinity-aware throughput maximization problem. We also proposed a heuristic for the soft-affinity-aware throughput maximization problem, by devising an efficient approximation algorithm with provable approximation ratio for its special case and then extending that approximation algorithm to solve the original problem. For the online hard-affinity-aware throughput maximization problem, we devised an online algorithm with a good competitive ratio. We finally evaluated the performance of the proposed algorithms by simulations and implementations in a real test bed. Experimental results showed that the performance of the proposed algorithms are promising.

The future work of the locality- and hardware affinity-aware VNF placement includes (1) considering the fairness on the usage of GPU resources in cloudlets in mobile edge clouds. With the quick deployment of GPU resources in mobile edge clouds, GPUs may be shared by users just as CPUs. Given different priorities and payments of users, how to design a fairness-aware VNF placement algorithm is challenging; and (2) there may exist many different types of GPU resources, such as conventional GPUs, FPGAs, and neural network accelerators in mobile edge clouds. Their processing capacities can also be different in accelerating the execution of VNFs. How to consider such resource heterogeneity in the VNF placement is is another potential future work along this research line.

Acknowledgments

The authors would like to thank the three anonymous referees and the associate editor for their expertise comments and constructive suggestions, which have helped them improve the quality and presentation of the article greatly. The work of Zichuan Xu and Qiufen Xia was supported in part by the National Natural Science Foundation of China under Grant 61802048 and Grant 61802047, in part by the fundamental research funds for the central universities in China under Grant DUT19RC(4)035, DUT-RU Co-Research Center of Advanced ICT for Active Life, and the "Xinghai Scholar Program" in Dalian University of Technology, China. The work of Weifa Liang was supported by the Australian Research Council Discovery Project under Grant DP200101985. The work of Pan Zhou was supported in part by NSFC under Grant 61972448.

References

[1] S. Agarwal, F. Malandrino, C. Chiasserini, and S. De, "Joint VNF placement and CPU allocation in 5G," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 1943–1951.
[2] W. Bao, D. Yuan, B. B. Zhou, and A. Zomaya, "Prune and plant: Efficient placement and parallelism of virtual network functions," *IEEE Trans. Comput.*, vol. 69, no. 6, pp. 800–811, Jun. 2020.
[3] N. Buchbinder and J. Naor,"The design of competitive online algorithms via a primal-dual approach," *Found. Trend Theor. Comput. Sci.*, vol. 3, no. 2/3, pp. 93–263, 2007.
[4] R. Behravesh, E. Coronado, D. Harutyunyan and R. Riggio, "Joint user association and VNF placement for latency sensitive applications in 5G networks," in *Proc. IEEE 8th Int. Conf. Cloud Netw.*, 2019, pp. 1–7.
[5] F. Carpio, S. Dhahri, and A. Jukan, "VNF placement with replication for Loac balancing in NFV networks," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–6.
[6] Y. Chen and J. Wu, "NFV middlebox placement with balanced set-up cost and bandwidth consumption," in *Proc. 47th Int. Conf. Parallel Process.*, 2018, Art. no. 14.
[7] R. Cohen, L. Eytan, J. Naor, and D. Raz, "On the effect of forwarding table size on SDN network utilization," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 1734–1742.
[8] S. Zhang and Q. Zhang, "Sector: TCAM space aware routing on SDN," in *Proc. 28th Int. Teletraffic Congr.*, 2016, pp. 216–224.
[9] R. Cohen, L. Eytan, J. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 1346–1354.
[10] X. Fei, F. Liu, H. Xu, and H. Jin, "Towards load-balanced VNF assignment in geo-distributed NFV infrastructure," in *Proc. IEEE/ACM 25th Int. Symp. Quality Service*, 2017, pp. 1–10.
[11] GÉANT. Accessed: Mar. 2020. [Online]. Available: http://www.geant.net
[12] Accessed: Feb. 2020. [Online]. Available: http://www.cc.gatech.edu/projects/gtitm/
[13] R. Guerzoni *et al.*, "Network functions virtualization: An introduction, benefits, enablers, challenges and call for action," in *Proc. SDN OpenFlow World Congr.*, 2012, no. 1, pp. 1–16.
[14] Hewlett-Packard Development Company, "L.P. Servers for enterprise – bladeSystem, rack & tower and hyperscale," 2015. [Online]. Available: http://www8.hp.com/us/en/products/servers/
[15] H. Huang, S. Guo, J. Wu, and J. Li, "Service chaining for hybrid network function," *IEEE Trans. Cloud Comput.*, vol. 7, no. 4, pp. 1082–1094, Oct.–Dec. 2019.
[16] H. Huang, P. Li, and S. Guo, "Traffic scheduling for deep packet inspection in software-defined networks," *Concurrency Comput., Pract. Experience*, vol. 29, no. 16, 2016, Art. no. e3967.
[17] M. Huang, W. Liang, Z. Xu, W. Xu, S. Guo and Y. Xu, "Dynamic routing for network throughput maximization in software-defined networks," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.
[18] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.
[19] 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.
[20] Y. Li, L. T. X. Phan, and B. T. Loo, "Network functions virtualization with soft real-time guarantees," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.
[21] X. Li, X. Wang, F. Liu, and H. Xu, "DHL: Enabling flexible software network functions with FPGA acceleration," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst.*, 2018, pp. 1–11.
[22] T. Lukovszki and S. Schmid, "Online admission control and embedding of service chains," in *Proc. Int. Colloquium Structural Inf. Commun. Complexity*, 2015, pp. 104–118.
[23] Y. Ma, W. Liang, J. Wu and Z. Xu, "Throughput maximization of NFV-enabled multicasting in mobile edge cloud networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 2 pp. 393–407, Feb. 2020.
[24] J. Martins *et al.*"ClickOS and the art of network function virtualization," in *Proc. 11th USENIX Conf. Netw. Syst. Des. Implementation*, 2014, pp. 459–473.

[25] N. McKeown *et al.*,"OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2017.

[26] Y. Nam, S. Song, and J. Chung, "Clustered NFV service chaining optimization in mobile edge clouds," *IEEE Commun. Lett.*, vol. 21, no. 2, pp. 350 –353, Feb. 2017.

[27] S. Palkar *et al.*, "E2: A framework for NFV applications," in *Proc. 25th Symp. Operating Syst. Princ.*, 2015, pp. 121–136.

[28] D. W. Pentico,"Assignment problems: A golden anniversary survey," *Eur. J. Operational Res.*, vol. 176, no. 2, pp. 774–793, 2007.

[29] P. Raghavan and C. D. Tompson,"Randomized rounding: A technique for provably good algorithms and algorithmic proofs," *Combinatorica*, vol. 7, no. 4, pp. 365–374, 1987.

[30] H. Ren *et al.*, "Efficient algorithms for delay-aware NFV-enabled multicasting in mobile edge clouds with resource sharing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 9, pp. 2050–2066, Sep. 2020.

[31] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 2–16, Feb. 2004.

[32] Y. Song, S. S. Yau, R. Yu, X. Zhang, and G. Xue, "An approach to QoS-based task distribution in edge computing networks for IoT applications," in *Proc. IEEE Int. Conf. Edge Comput.*, 2017, pp. 32–39.

[33] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Efficient algorithms for capacitated cloudlet placements," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 10, pp. 2866–2880, Oct. 2016.

[34] Z. Xu, Z. Zhang, W. Liang, Q. Xia, O. F. Rana, and G. Wu, "QoS-Aware VNF placement and service chaining for IoT applications in multi-tier mobile edge networks," *ACM Trans. Sensor Netw.*, vol. 16, no. 3, Jun. 2020, Art. no. 23.

[35] VXLAN. Accessed: Jun. 2020. [Online]. Available: https://tools.ietf.org/html/rfc7348

[36] B. Yang, W. Chai, G. Pavlou, and K. V. Katsaros, "Seamless support of low latency mobile applications with NFV-enabled mobile edge-cloud," in *Proc. 5th IEEE Int. Conf. Cloud Netw.*, 2016, pp. 136–141.

[37] B. Yang *et al.*, "Algorithms for fault-tolerant placement of stateful virtualized network functions," in *Proc. IEEE Int. Conf. Commun.*, 2018, pp. 1–7.

[38] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. Workshop Mobile Big Data*, 2015, pp. 37–42.

[39] X. Yi *et al.*, "FlowShader: A generalized framework for GPU-accelerated VNF flow processing," in *Proc. IEEE 27th Int. Conf. Netw. Protocols*, 2019, pp. 1–12.

[40] R. Yu, G. Xue, and X. Zhang, "QoS-aware and reliable traffic steering for service function chaining in mobile networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2522–2531, Nov. 2017.

[41] Z. Xu, W. Liang, A. Galis, Y. Ma, Q. Xia and W. Xu,"Throughput optimization for admitting NFV-enabled requests in cloud networks," *Comput. Netw.*, vol. 143, no. OCT. 9, pp. 15–29, 2018.

[42] X. Yi, J. Duan and C. Wu, "GPUNFV: A GPU-accelerated NFV system," in *Proc. 1st Asia-Pacific Workshop Netw.*, 2017, pp. 85–91.

[43] K. Zhang *et al.*, "G-NET: Effective GPU sharing in NFV systems," in *Proc. 15th USENIX Conf. Netw. Syst. Des. Implementation*, 2018, pp. 187–200.

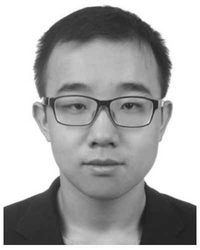
[44] Q. Zhang, Y. Xiao, F. Liu, J. C. S. Lui, J. Guo, and T. Wang, "Joint optimization of chain placement and request scheduling for network function virtualization," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 731–741.

[45] Z. Zheng, J. Bi, C. Sun, H. Yu, H. Hu, and Z. Meng, "GEN: A GPU-accelerated elastic framework for NFV," in *Proc. 2nd Asia-Pacific Workshop Netw.*, 2018, vol. 143, pp. 57–64.

[46] P. Vizarreta, M. Condoluci, C. Machuca Mas, T. Mahmoodi and W. Kellerer, "QoS-driven function placement reducing expenditures in NFV deployments," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–7.

[47] B. Yang, W. Chai, G. Pavlou, and K. V. Katsaros, "Seamless support of low latency mobile applications with NFV-enabled mobile edge-cloud," in *Proc. 5th IEEE Int. Conf. Cloud Netw.*, 2016, pp. 136–141.

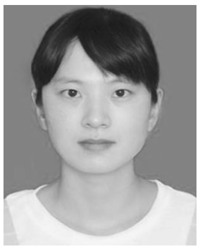
Zichuan Xu (Member, IEEE) received the BSc and ME degrees from the Dalian University of Technology, China, in 2008 and 2011, respectively, both in computer science, and the PhD degree from Australian National University, Australia, in 2016. He was a research associate at the Department of Electronic and Electrical Engineering, University College London, United Kingdom. He currently is an associate professor at 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.



Zhiheng Zhang received the BSc and MSc degrees from the School of Software, Dalian University of Technology, Dalian, China, in 2017. He is currently working toward the PhD degree from the School of Software, Dalian University of Technology, Dalian, China. His major research includes software define network (SDN), network function virtualization (NFV), resource management for mobile edge cloud, and block chains.

John C. S. Lui (Fellow, IEEE) is currently the Choh-Ming Li chair professor at the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. His current research interests include Internet, network sciences with large data implications, machine learning on large data analytics, network/system security, network economics, large scale distributed systems and performance evaluation theory. He received various departmental teaching awards and the CUHK Vice-Chancellor's Exemplary Teaching Award. He also received the CUHK Faculty of Engineering Research Excellence Award (20112012). He is a co-recipient of the best paper award in the IFIP WG 7.3 Performance 2005, IEEE/IFIP NOMS. He is also a fellow of the ACM.

Weifa Liang (Senior Member, IEEE) received the BSc degree from Wuhan University, China, in 1984, the ME degree from the University of Science and Technology of China, China, in 1989, both in computer science, and the PhD degree from Australian National University, Australia, in 1998. He is currently a full professor at the Research School of Computer Science, Australian National University, Australia. His research interests include design and analysis of energy efficient routing protocols for wireless ad hoc and sensor networks, Internet of Things(IoT), mobile edge computing and cloud computing, Network Function Virtualization (NFV) and Software-Defined Networking (SDN), design and analysis of parallel and distributed algorithms, approximation algorithms, combinatorial optimization, and graph theory.



Qiufen Xia (Member, IEEE) received the BSc and ME degrees from the Dalian University of Technology, China, in 2009 and 2012, respectively, both in computer science, and the PhD degree from Australian National University, Australia, in 2017. She is currently a lecturer at the International School of Information Science and Engineering, Dalian University of Technology, China. Her research interests include big data processing, mobile cloud computing, distributed clouds, cloud computing, and optimization problems.

Pan Zhou (Senior Member, IEEE) received the BSc degree in the advanced class of the Huazhong University of Science and Technology (HUST), China, and the ME degree in electrical engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2006 and 2008, respectively. He also received the MS degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, Georgia in 2010, and the PhD degree from the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, Georgia, in 2011 as a member of the Communications Systems Center (CSC), Atlanta, Georgia. He held honorary degree and merit research award of HUST in his master study. He was once a student internship at Mobile Communications and Networking Research, NEC Laboratories America, Inc., in summer 2010. His research interest includes: wireless networks, security and privacy, and Big Data analytics.

Wenzheng Xu (Member, IEEE) received the BSc, ME and PhD degrees in computer science from Sun Yat-Sen University, Guangzhou, PR China, in 2008, 2010, and 2015, respectively. He currently is a special associate professor at the Sichuan University, China, and was a visitor at the Australian National University, Australia. His research interests include wireless ad hoc and sensor networks, mobile computing, approximation algorithms, combinatorial optimization, online social networks, and graph theory.

Guowei Wu received the PhD degree from Harbin Engineering University, PR China, in 2003. He is currently a professor at the School of Software, Dalian University of Technology (DUT), China. His research interests include embedded real-time system, cyber-physical systems (CPS), and smart edge computing. He has published more than 100 journal and conference papers.