Computation Offloading via Sinkhorn's Matrix Scaling for Edge Services

Lei Wang, Yunqiu Zhang, and Shuhan Chen

***Abstract*—Mobile-edge computing (or MEC) has aroused a wide attention at the 5th generation mobile networks (5G) communication era. Different edge servers (such as cloudlets, micro data centers, and base stations) have been proposed to support the MEC architecture paradigm. For the diverse loading capacities of different edge servers, computation offloading for the edge services loaded in neighboring edge servers is desired for assuring the overall serve performance as well as the Quality of Experience of users for MEC applications. To dynamically balance the computation load for neighboring edge servers, we model the optimal edge services computation offloading destination determination issue as an optimal transport distances problem in this article. We propose computation offloading via Sinkhorn's matrix scaling (COSIMS) to determine the optimal offloading destination. Experimental evaluations conducted on real-world edge computing data sets indicate that COSIMS can guarantee that the neighboring edge servers cooperatively provide effective services to mobile users with least extra communication hops.**

***Index Terms*—Computation offloading, edge services, optimal transport (OT) distances, Quality of Experience (QoE), Sinkhorn's matrix scaling.**

I. Introduction

With the rapid development of 5G mobile communication and Internet-of-Things (or IoT) technologies, mobile-edge computing (or MEC) has aroused great interests for performant resource-intensive smart mobile applications [1]–[4]. Again leading the computing paradigm to distributed computing, MEC aims at tackling the challenges of response latency and data storage of the centralized-only cloud computing architecture paradigm [5], [6].

For the promising application of MEC, Ad Hoc mobile devices cannot provide sufficient computation capabilities [7]. In contrast, new scenarios and applications, such as virtual reality/augmented reality (VR/AR), real-time video analytics, and autonomous driving, etc., desire for a data incentive local processing ability of the client-side mobile devices [4], [5], [8]. Toward a better user Quality of Experiences (QoEs) of MEC applications, a fast deployed customization service, namely, edge service, could be deployed to serve the neighboring mobile devices [9]. Deployed in edge server, edge service aims at offloading the computing load of the terminal devices and henceforth mitigating the response delay caused by data communication with the remote center cloud [10].

Manuscript received June 9, 2020; revised November 10, 2020; accepted November 30, 2020. Date of publication December 3, 2020; date of current version May 7, 2021. This work was supported in part by the Humanity and Social Science Fund of MoE of China under Grant 18YJCZH170, and in part by the 2019 Six Talent Peaks Project in Jiangsu Province under Grant RJFW-029. *(Corresponding author: Lei Wang.)*

The authors are with the Department of Management Science and Engineering, Nanjing Forestry University, Nanjing 210037, China (e-mail: leiwangchn@163.com).

Digital Object Identifier 10.1109/JIOT.2020.3042164

In a (VR/AR)-oriented MEC application, the edge server can be deployed near the mobile base station and communicates with the base station by wired or wireless network. As for the business model, the edge server can be hosted and maintained by mobile network operators [11]. By renting the virtual machines (VMs) in the edge computers from mobile operators, MEC application providers can deploy edge services in the VMs to support the application needs of nearby mobile devices. There are different types of edge servers that can support edge services operation [3], e.g., cloudlets [9], micro data centers (micro DCs) [12], [13], and base stations [14], [15], etc.

Different edge servers are hosted by different mobile operators. For different internal hardware and software equipment and the external execution environment, the computing abilities of different edge servers are diverse. The real-time loads of different edge servers are also different. For the real-time application requirement, offloading the computation tasks for edge services should also consider the number of communication hops from the mobile devices to the destination edge server. To guarantee the QoE of the users for an MEC application, it is a challenging problem to offload computation tasks of the edge services between neighboring edge servers.

Relevant existing computation offloading approaches are insufficient in tackling the novel challenges that arise in computation offloading for edge services and considering the diversity of loading abilities of different edge servers and the overall computation load balancing as well as the additional communication cost [16]–[19]. Specifically, to minimize execution time and energy consumption of mobile devices' computing tasks, computation offloading from mobile devices to the edge services or center cloud has been widely investigated [20], [21]. The number of network communication hops and the offloading costs are considered [17], [22]. To balance the load between edge servers, integer programming and particle swarm optimization (PSO) were employed for computation offloading among edge servers [18], [19].

In contrast to the existing works, we take the diversity of edge servers, the load balance and the additional network communications hops together into consideration for edge service computation offloading in this article. A novel computation offloading via Sinkhorn's matrix scaling (or COSIMS) approach is proposed. The approach serves the purpose of raising the overall performance of mobile users and reducing the computation complexity for edge service computation offloading.

Comparing with the existing integer programming and metaheuristic-based edge service computation offloading approaches, the objective function defined based on entropy regularization in COSIMS approach is a convex function. COSIMS solves the optimal offloading destination via matrix scaling. The objective function defined in the existing works is not guaranteed to be a convex function. It is easy to fall into the local optimal solution. In practice, it is difficult to guarantee the optimization effect by the existing approaches. The Sinkhorn algorithm has a strict mathematical theory foundation. For the problem of edge service computation offloading, COSIMS approach can ensure the load balance of various edge servers and minimize the number of extra communication hops. We summarize the following contributions.

1) We model the edge service computation offloading as an optimal transport (OT) distance problem. Benchmark computing power (BCP) is defined to measure the computation processing abilities of diverse edge servers. A unified resource utilization ratio value RU is determined to balance loads of different edge servers. The optimal computation offloading matrix is defined to minimize the offloading cost and balance the loads.
2) We propose COSIMS based on Sinkhorn's matrix scaling to solve the optimal edge service computation offloading problem. The optimization issue is regularized by minimizing the entropy of the offloading matrix to approach the optimal solution. The fixed-point iteration approach is employed to solve the optimization issue. A VR/AR-oriented edge computing architecture is presented to reify the application of COSIMS.
3) We conducted large-scale simulation experiments under real-world edge computing EUA repository data sets. The experimental results show that COSIMS is better than density-based cluster (DBC), Kantorovich linear programs (KLPs) and Random assignment approaches in terms of extra cost ratio (ECR), performance gain (PG), and average convergence time—indicating that COSIMS appears better optimization effect and lower computational complexity.

The remainder of this article is organized as follows. We present a motivating example and the formal definitions of the studied problem in Section II. We introduce COSIMS in detail in Section III. We give the implementation of the experiments in Section IV. We summarize the important related works in Section V. We conclude by identifying some future directions in Section VI.

II. Motivation and Formalization

A. Motivating Example

As illustrated in Fig. 1, let us begin by presenting a (VR/AR)-oriented MEC application. Game players as the users participate in the gunfight game via their mobile equipments, i.e., headwear equipments and handheld devices. Location and

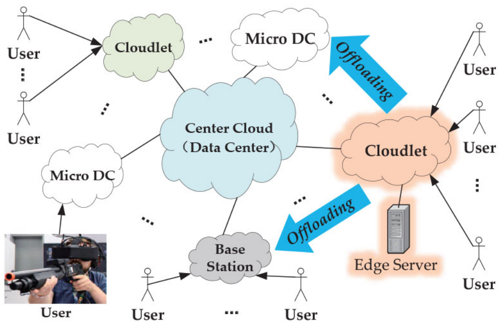

Fig. 1. VR/AR MEC application.

action sensors in the handheld device identify the activities of each player. Headwear equipment performs 3-D image rendering and displays the results to the player. All the equipments communicate with the center cloud to dynamically update the map and the game scenario in real time.

Vertigo is a serious problem for the state-of-the-art VR/AR applications. The vertigo problem is mainly attribute to the response delay of 3-D images processing. Under the 5G network, the speed of data communication is greatly enhanced. For example, the video delivery and online video can achieve sound application effects. The main bottleneck of VA/AR applications leading to response delay is the low computation ability of 3-D image rendering of the mobile headwear equipments. We need to migrate the computing tasks of headwear equipments to remote center clouds. However, this will result in a communication delay due to multiple hops of the remote network communications.

To mitigate the response delay problem, we can set up different types of edge servers near the users. Customized edge services for the VA/AR game can be deployed in the neighboring edge servers. Each edge computer has a high 3-D image processing ability. It communicates with the center cloud to synchronize data. Computation tasks in headwear equipments can therefore be offloaded to edge services at close range with lesser hops in the network communication.

If computing tasks of the edge services are concentrated on an individual edge server (e.g., the cloudlet edge server), it will cause delay for the corresponding mobile application. There desires for an effective optimization approach for edge services allocation to offload the computation tasks in overloaded edge servers to the neighboring edge servers (e.g., the micro DC and base station).

B. Problem Formalization

In this section, we present the concepts of computation offloading for edge services and give the formal definition of optimal computation offloading problem. Common terms used in this article are summarized in Table I.

Definition 1 (Benchmark Computing Power): We define τ as the BCP for a software cloud service, i.e., τ represents the ability to complete a benchmark task (e.g., 1000 atomic arithmetic or logic operations) under a fixed period (e.g., 1 ms).

TABLE I
TERMS IN COMMON USE

Terms	Descriptions
Mobile Edge Computing	A new computing paradigm.
Edge Service	A service deployed at the edge.
Center Cloud	Data center at the cloud.
Edge Server	A computer at the edge.
Cloudlet	A type of Edge Server.
Micro Data Center	A type of Edge Server.
Base Station	A type of Edge Server.
Benchmark Computing Power	A unit of computing power.
Offloading Domain	Neighboring edge servers.
Offloading Polytope	Offloading schemes matrices.
Offloading Cost Matrix	A pairwise cost matrix.

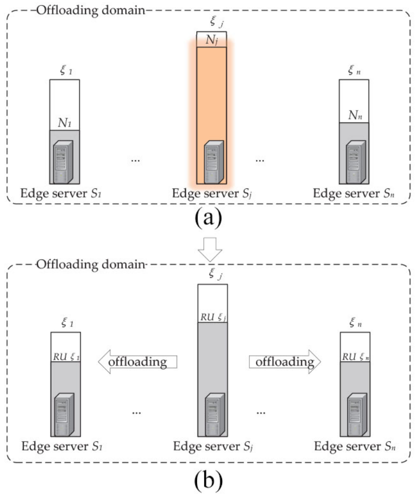

Fig. 2. Optimal offloading objective for each edge server. (a) Received service requests. (b) Optimal offloading objective.

As a software service which can be invoked by the mobile devices, each edge service C_i is deployed on an edge server S_j. In this article, different forms of edge cloud containers, including cloudlets, micro DCs and base stations are all defined as edge servers. The edge server assigns a VM for each edge service. We assume that each VM is assigned with a BCP of τ by the edge server.

Definition 2 (Offloading Domain): We define an offloading domain for the edge servers as ℓ, for which each included edge server is nearby to each other. Edge services in any two edge servers in the offloading domain can be offloaded to one another. Any two edge servers can directly access each other within at most ψ network communication hops.

As illustrated in Fig. 2(a), since different types of edge servers have different computation processing abilities; let the overall BCP for any edge server S_j in the offloading domain be ξ_j, for $j \in \{1, 2, \ldots, n\}$, we assume that there are altogether N_j edge services request for S_j. We define the global unified resource utilization ratio value RU as

$$\mathrm{RU} = \frac{\sum_{j=1}^{n} N_j}{\sum_{j=1}^{n} \xi_j}. \tag{1}$$

If we allocate the edge services to each of the edge server S_j according to the globally unified resource utilization ratio value RU and the overall BCP of ξ_j, all the edge servers will achieve an equally resource utilization ratio and the optimal performance. Hence, for each edge server S_j, the optimal number of loaded edge services is RUξ_j. We set RUξ_j as the optimal offloading objective for each edge server [see Fig. 2(b)].

We will obtain two probability column vectors to represent the received service requests and the optimal offloading objective for all the edge servers in the offloading domain, respectively, as

$$r = \left\{ r_i \in \mathbb{R}_+^n : \sum_{i=1}^{n} r_i = 1 \right\} \tag{2}$$

and

$$o = \left\{ o_i \in \mathbb{R}_+^n : \sum_{i=1}^{n} o_i = 1 \right\} \tag{3}$$

where $i \in \{1, 2, \ldots, n\}$ represents the number of the edge servers in the offloading domain, $r_i = [N_i/(\sum_j N_j)]$ and $o_i = [\xi_i/(\sum_j \xi_j)]$ represent the probability of the number of edge services assigned to the edge server S_i for the received service requests and optimal offloading objective, respectively.

To offload the edge services which may lead to potential computation overload of edge servers, we need to solve the problem of how to transform the vector r to o. This can be cast as an OT distance problem [23]. We will refer to the OT to model the problem.

Definition 3 (Offloading Polytope): We define the edge service offloading polytope $U(r, o)$ as $n \times n$ matrices, where

$$U(r, o) = \left\{ P \in \mathbb{R}_+^n | P 1_d = r, P^T 1_d = o \right\} \tag{4}$$

where $P[i, j]$ is a proportion value and $\lfloor P[i, j] \times \sum_j N_j \rfloor$ is the number of edge services in edge server r_i that should be offloaded to edge server o_j, for $i, j \in \{1, 2, \ldots, n\}$.

$U(r, o)$ is not unique and each represents a possible offloading strategy. For each edge service, if we offload it to a farther edge server (the distance is judged based on the number of communication hops), it will lead to a bigger offloading cost; resulting in a larger communication delay. Hence, the optimal computation offloading for the edge services should make overall offloading cost be the least. We define the following offloading cost matrix.

Definition 4 (Offloading Cost Matrix): Assume that the number for the communication hops from the mobile device to the edge server which received the service request be 1, we define an offloading cost matrix M, in which the pairwise cost of $M[i, j] \in [1, \psi]$ represents the number of communication hops when edge server S_i accesses the edge server S_j, where the communication hops are determined according to the optimal network communication routing.

To offload the computation tasks and guarantee the overall performance of MEC systems, we need to offload the edge service in the edge server to make all the edge servers satisfy the optimal offloading objective. Meanwhile, we should also guarantee that the overall offloading cost would be least. Similar to the OT problem, we adopt the Frobenius dot product (i.e., $\langle \cdot, \cdot \rangle$) to calculate the overall offloading cost. The optimal edge services offloading strategy would be determined by the following.

Definition 5 (Optimal Computational Offloading Matrix): Given an offloading cost matrix M ($M \in \mathbb{R}_{+}^{n\times n}$), the optimal computation offloading matrix to offload the edge services from the received edge service requests r to the optimal offloading objective o is defined as P^*, which is obtained by solving the following problem:

$$P^* = \underset{P\in U(r,o)}{\arg\min} \langle P, M\rangle \tag{5}$$

where

$$\langle P, M\rangle = \sum_{i,j\in\{1,2,\ldots,n\}} P[i,j] \times M[i,j]. \tag{6}$$

In particular

$$d_M(r,o) = \left\langle P^*, M\right\rangle \tag{7}$$

is the optimal overall offloading cost.

It is worth noting that, by solving the optimal computation offloading problem, we can guarantee that each edge server within the offloading domain will work at a globally unified resource utilization ratio. The problem of computation overload in individual edge server can be mitigated. The additional communication cost for the edge services is minimal. The overall performance of the system would be enhanced.

III. COSIMS Approach

In this section, we present COSIMS to determine the optimal computation offloading matrix P^* for MEC applications. Different optimization approaches can be used to solve this problem. Sinkhorn's matrix scaling approach has sound mathematical foundation of probability theory, and can guarantee the optimization effectiveness of the results by defining a convexity objective function. Therefore, COSIMS is designed based on Sinkhorn's matrix scaling approach.

A. Entropy Regularization

The OT problem has been widely investigated in data sciences, such as contrast equalization, texture synthesis, data drift detection, etc [24].

As for any offloading matrix $P \in U(r,o)$, each value of $P[i,j]$ indicates the proportion of edge services in edge server i that should be offloaded to edge server j. More general, the value of $P[i,j]$ can be cast as a joint distribution for two multidimensional random variables, i.e., $X = i, Y = j$, and $P[i,j] \in P(X,Y)$, where the values of $i,j \in \{1,2,\ldots,n\}$. We define $H(P)$ as the entropy of matrix P as

$$H(P) = -\sum_{i,j\in\{1,2,\ldots,n\}} P[i,j]\log P[i,j] \tag{8}$$

in which, a larger value of $H(P)$ indicates a more sparse matrix of P. In contrast, the matrix would be smoother.

To determine P^*, we first define a Lagrange multiplier $\lambda \in [0,1]$ to approximate the optimal solution. We define the following regularized optimal computation offloading matrix as:

$$\begin{aligned} P^\lambda &= P^* - \lambda \times H(P) \\ &= \underset{P\in U(r,o)}{\arg\min} \langle P, M\rangle - \lambda \times H(P). \end{aligned} \tag{9}$$

In particular, with the parameter λ, the overall offloading cost

$$d_M^\lambda(r,o) = \langle P^\lambda, M\rangle, \left(d_M^\lambda(r,o) \geq d_M(r,o)\right) \tag{10}$$

is defined as a dual-Sinkhorn divergence [25]. If we offload each r_i according to the loading ability of each of the optimal offloading objective edge server as o_i, we will get a smooth offloading matrix ro^T. Here, ro^T can be cast as the worst case of optimal offloading. In this case, we can only adjust the value of λ to compute P^λ to approximate P^*, and the searching range is between ro^T and P^*.

To summarize, when we fix the value of λ, a lower value of $H(P^\lambda)$ will make the matrix P^λ more sparse, the P^λ more approach to P^*. This is a convex optimization problem. In practice, we can reduce the value of λ to make the P^λ more approach to P^*. We will adopt Sinkhorn's matrix scaling algorithm to solve the optimization problem.

B. Matrix Scaling

In this section, we introduce how to reduce the value of $H(P^\lambda)$ based on Sinkhorn's matrix scaling and present COSIMS algorithm.

Let u, v be two nonnegative vectors of $\mathbb{R}_{+}^{n}$, we define two diagonal matrixes $\text{diag}(u)$ and $\text{diag}(v)$, and a kernel matrix K to scale matrix P^λ within the range of $U(r,o)$ based on Sinkhorn's diagonal equivalence theorem [26], [27] as

$$P^\lambda = \text{diag}(u)K\text{diag}(v) \tag{11}$$

where

$$K = \exp\left(-\frac{1}{\lambda}M\right). \tag{12}$$

Given the kernel matrix $K \in P$ and r, o, COSIMS algorithm updates the value of P^λ by Sinkhorn's fixed-point iteration approach. A sufficient number of times of alternative updates of u and v will make the regularized optimal computational offloading matrix converge to P^λ. The fixed-point iteration can be executed as

$$u = \frac{r}{Kv} \tag{13}$$

and

$$v = \frac{o}{K^T u}. \tag{14}$$

This fixed-point iteration can therefore be expressed as a single update

$$u = \frac{r}{K\frac{o}{K^T u}}. \tag{15}$$

To solve the computation offloading problem, we will first initialize the value of u. Then, we update the u value by the fixed-point iteration [i.e., (15)] until u will unchange or reach the maximal iteration number. Next, we determine the value of v using (14). Finally, we will obtain P^λ by substituting the values of u and v into (11). The algorithm for determining the P^λ is named COSIMS and is summarized in Algorithm 1.

It is worth noting that, if COSIMS algorithm is not executed at the first time in an MEC system, some received edge service requests may have been offloaded to other edge servers. The

Algorithm 1 COSIMS

```
Input: $M \in \mathbb{R}_+^{n\times n}$, $\lambda$, $r$, $o$, $MaxIteration$
Output: $P^\lambda$
1: Initialize $K$ by solving Eq. (12);
2: Initialize $u = [1/n, 1/n, \cdots, 1/n]_n$;
3: while $d \in [1, MaxIteration]$ and $u$ changes do
4:     update $u$ by solving Eq. (15);
5:     $d = d + 1$;
6: end while
7: compute $v$ by solving Eq. (14);
8: compute $P^\lambda$ by solving Eq. (11);
9: return $P^\lambda$;
```

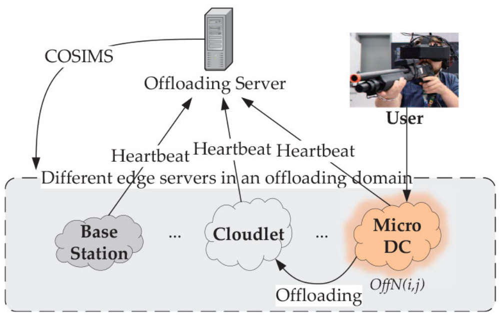

Fig. 3. Reifying application of COSIMS.

actual loaded edge services in the edge servers may not equal to r. To guarantee the effectiveness of COSIMS algorithm, we consistently use the received edge service request r as the input for each time COSIMS algorithm executes. In this way, we do not need to update the offloading cost matrix M during the algorithm execution.

Using the regularized optimal computation offloading matrix P^λ, we will have the offloading scheme for the edge services. For any two edge servers S_i and S_j ($i, j \in \{1, 2, \ldots, n\}$), the number of edge services that should be offloaded from S_i to S_j will be

$$\text{Off}N(i,j) = \left\lfloor P^\lambda[i,j] \times \sum_j N_j \right\rfloor. \tag{16}$$

C. Reifying Through Application

In this section, we clarify the application of COSIMS for VR/AR Edge Computing. As illustrated in Fig. 3, neighboring edge servers are identified as an offloading domain. Different types of edge servers, including base station, cloudlet, and micro DC, may be included in the offloading domain. According to Definition 2, the communication hops between any two edge servers should not exceed ψ. The maximal number of edge servers from the source to the objective is $\psi + 1$ in an offloading domain.

An offloading server is deployed in the offloading domain. The offloading server collects the real-time number of total received service requests of each included edge server by a heartbeat mechanism. In practice, the offloading server can be realized as an edge service and deployed in an edge server. For example, it can be deployed in a special used micro DC, or in a general purpose used cloudlet or base station. In the positions where there are many end users and edge servers distributing around, an offloading server can be deployed there.

Each user of the VR/AR application is assigned to the nearest edge server based on the distance from the user to edge server. An edge service will be realized in the connected edge server to execute the user's computing tasks. The offloading server executes COSIMS algorithm periodically to determine the optimal computation offloading destination for overloaded edge servers. For each edge server i, Off$N(i, j)$ will be determined based on COSIMS–indicating that the computing tasks for the number of Off$N(i, j)$ edge services should be offloaded to edge server j. Suppose that a user is connecting to the edge service i (i.e., micro DC), the computation task for the number of edge services Off$N(i, j)$ will be offloaded to the edge server j (i.e., cloudlet). To guarantee the overall performance of the VR/AR system and the QoE of all users, we can randomly select the edge services in edge server i and offload them to the edge server j. By COSIMS, all the edge servers in the offloading domain will achieve a uniform resource utilization ratio value RU. Thus, the load of the edge servers will be balanced.

Note that no matter for different performance traits of different types of edge servers, such as different 3-D image rendering or I/O throughput abilities, we assume that each edge server assigns a VM with a BCP of τ for each edge service in this article. Hence, each edge service can be offloaded to different types of edge servers. In some specific applications, multiple mobile devices may have different edge computation resource requirements. Since different edge servers (i.e., cloudlet, micro DC, and base station) have different performance traits, the computation tasks of edge services should be offloaded to the same type of edge servers. The offloading domain should be determined by the same type of edge servers. In this case, we should calculate the required BCPs for each mobile device. Then, COSIMS will also effectively work for computation offloading among an offloading domain with the same type of edge servers.

To reduce the communication delay, when the edge services in the overloaded edge server i are offloaded to others, the user in the VR/AR application still accesses the closet edge server i. User requests are forwarded to the objective edge server j. Since in (2), the r represents the received edge service requests for the edge servers in the offloading domain, COSIMS algorithm only inputs r as the number of computation tasks for each edge server to determine the offloading destination. If the offloading destination of an edge service is different from the current edge server the edge service is working on, the edge service will be migrated to the new destination.

To migrate the ongoing edge services to the destination edge servers, the VM live migration techniques are more fit for this article [28]. Specifically, since the edge servers host some intermediate data processing results, such as the users' state information, we need to perform a stateful migration for the edge services [29]. To guarantee the consistency of the data and the real-time performance of the application, we present the following steps for the edge service migration based on precopy memory data migration. First, we start a process in the source edge server to forward messages to the destination edge service. Second, we assign a VM in the destination edge server and deploy an edge service in the VM for the application of the mobile device. Third, we duplicate data from the source to the destination and then switch to the destination edge service as the working edge service. Finally, we recover the migrated VM resource in the source edge server.

IV. Experimental Study

We conducted the following experiments to verify the effectiveness and efficiency of COSIMS. The data sets in EUA repository [30] were used to simulate the edge servers and the user requests. We compared COSIMS with three representative approaches. All the experiments were conducted in Python on a PC equipped with Windows 7 × 86 Enterprise Edition OS, and Intel Core i7 2600 CPU, 12-GB RAM, Seagate 1-TB HDD.

A. Approaches Under Comparison

To study the effectiveness of COSIMS for edge service computation offloading, we compared our approach with the relevant existing edge service placement approach and the approach used for solving optimal transportation problem. The three approaches under comparison include

1) *Density-Based Cluster:* Based on [31], when a user request arrives, we assign an edge service to the nearest edge server to serve for the user. For each edge server j, when the assigned edge server reaches RUξ_j, the subsequent user requirements will be forwarded to the nearest edge server with least extra communication hops.
2) *Kantorovich Linear Programs:* We employ the KLP approach to determine the optimal offloading matrix P^* based on the approach presented in [24].
3) *Random Assignment (Random):* We randomly select the offloading objective in the offloading domain. We also guarantee a unified resource utilization ratio value RU during the offloading process.

B. Metrics

We define the following metrics to evaluate the effectiveness and efficiency of the approaches.

1) *Extra Cost Ratio:* We use the number of extra communication hops to evaluate the offloading cost. Let C_r and C_o be the overall number of communication hops for all mobile users before and after the computation offloading, respectively. We define ECR as

$$\mathrm{ECR} = \frac{C_o - C_r}{C_r} \times 100\%. \tag{17}$$

2) *Performance Gain:* Inspired by [32] and [33], we define PG as the metric to evaluate the potential overall performance improvement ratio. Since lower load of an edge server will result in more CPU time and memory usage for each edge service, let $\mathrm{Num}r_j$ and $\mathrm{Num}o_j$ represent the number of loaded edge services before and after the computation offloading, PG is defined as

$$\mathrm{PG} = \frac{\sum_j \frac{\xi_j}{\mathrm{Num}o_j} - \sum_j \frac{\xi_j}{\mathrm{Num}r_j}}{\sum_j \frac{\xi_j}{\mathrm{Num}r_j}} \times 100\% \tag{18}$$

where $\mathrm{Num}r_j \neq 0$, and $\mathrm{Num}o_j \neq 0$.
3) *Average Convergence Time:* We note the average execution time to compare the computational complexity of different computation offloading approaches.

TABLE II
Parameters Investigated

Parameters	Descriptions
λ	Lagrange multiplier.
ψ	The maximal communication hops.
Number of mobile users	To simulate the loads.
n	Node number.
$MaxIteration$	Maximal iteration times.

C. Setup

To simulate the MEC application environment, we used the data in EUA repository. The open EUA repository contains a set of real-world edge servers and edge users data sets collected in the Australia region. It is regarded as the benchmark data sets for edge computing researches [6], [10], [15]. The site.csv data set in EUA repository contains 95 562 base station data and can thus provide more data for simulating the edge servers. Thus, the site.csv and the corresponding user data users-aus.csv were used in this article to simulate the edge servers and the mobile users. We first randomly selected edge servers from the site.csv. Then we generated an Erdos–Renyi (or ER) random graph to simulate the routing table for the edge servers. Each node in the ER random graph is an edge server. Undirected edges in the ER random graph were used to represent the communication links. A shortest path routing was identified from the ER random graph. During the experiments, the maximal communication hops were all set as $\psi \leq 5$. The communication hops between any two nodes in the ER random graph were noted and then we obtained the offloading cost matrix M. We randomly selected users from the users-aus.csv data. Each user was assigned to the nearest edge server to simulate a user request according to the distances calculated based on the latitude and longitude of the edge servers and the user. Different computation offloading approaches were used to offload computation loads for the edge servers, respectively. The *MaxIteration* for COSIMS is set as 35. Each experiment was conducted 30 times and the averaged results were noted and analyzed. The parameters investigated during the experiments are summarized in Table II.

D. Impact of λ

In this experiment, we study how the value of Lagrange multiplier λ impacts the performance of COSIMS approach. We set the node number of ER random graph as 30 and the number of mobile users as 500. We varied the value of λ from 0.1 to 1, with a step value of 0.1. For each value of λ, we repeated the experiments 30 times by reselecting the edge

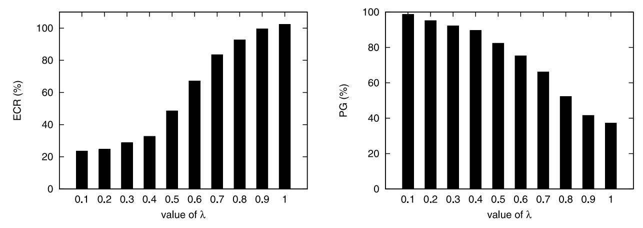

Fig. 4. Impact of Lagrange multiplier λ. (a) ECR. (b) PG.

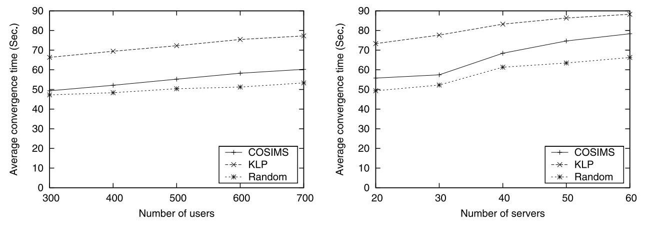

Fig. 6. Average convergence time of different approaches. (a) 25 edge servers. (b) 600 mobile users.

servers and mobile users randomly. The average ECR and PG are analyzed.

As shown in Fig. 4, the PG declines with the enlargement of the value of λ. This indicates that the effectiveness of COSIMS algorithm declines with the enlargement of λ. This is because when λ is smaller, P^{λ} is closer to the optimal computation offloading matrix P^{*} as calculated in (9). To deal with the MEC application response delay issue, we should set a smaller value of λ. The ECR values granularly increase with the increase of λ value. When $\lambda > 0.5$, the ECR values show more obvious enlargement. This indicates that COSIMS algorithm can guarantee smaller number of extra communication hops for MEC system application under the situation when the offloading effectiveness is guaranteed.

E. Performance Comparison

This experiment aims at comparing different approaches to analyze the performance of the COSIMS algorithm. As for the COSIMS algorithm, we fixed the Lagrange multiplier as $\lambda = 0.1$. We first fixed the number of edge servers by setting the nodes number of ER random graph as 25 and varied the number of mobile users from 300 to 700, with a step value of 100. Then, we fixed the number of mobile users as 600 and varied the number of edge servers from 20 to 60, with a step value of 10. The approaches of COSIMS, DBC, KLP, and Random were executed, respectively. The results of ECR and PG for different approaches are summarized in Fig. 5.

As can be seen from the results: 1) the ECR and PG values of different approaches increase with the increase of the number of mobile users and decrease with the increase of the number of edge servers. This is because when we fixed the number of edge servers, the increase of the number of mobile users will make that more edge services should be offloaded. The computation offloading operation enlarges the extra communication cost, but it achieves higher PG for all mobile users and 2) besides, for the DBC approach, the ECR values of the COSIMS approach are the lowest. The ECR values of the DBC method and COSIMS method are very close. The DBC approach directly offloads the overloaded edge services to the nearest edge servers, so the extra communication costs are lower than the COSIMS approach. However, the COSIMS method has the highest PG value among all the approaches in terms of PG. By COSIMS, the mobile devices first directly connect to the nearest edge server. If the nearest edge server is very busy, the computation task would be offloaded to the idle edge servers in the offloading domain. This will raise the number of commutation hops. But if the mobile devices directly access the remote idle edge server, mobile communication quality may be worse. Therefore, COSIMS can effectively avoid declines in communication quality. It can also effectively guarantee the computing performance of all mobile users. COSIMS is more practical for MEC applications.

F. Computational Complexity

The convergence time of different computation offloading algorithms is important for MEC applications, such as for delay-sensitive VR/AR applications. To investigate the efficiency of different approaches for real-time computation offloading application, we noted and summarized the average convergence time of COSIMS and the other approaches during the experiment conducted in Section IV-E. The results are shown in Fig. 6.

As can be seen from the results, the average convergence time of COSIMS is obviously lower than the KLP approach. This indicates that the Sinkhorn algorithm can achieve faster convergence than linear programs, while the optimal effect is better (retrieve Fig. 5). Although the convergence time of Random approach is lower than that of the COSIMS approach, the optimal effect in terms of ECR and PG is worse than COSIMS and Random approaches. As for DBC approach, since the offloading process is coupled with the user request process, we cannot calculate the convergence time for DBC under the same scale of COSIMS. In fact, the COSIMS approach executes the offloading operation at set intervals on the offloading server. The offloading operation does not affect the real-time response of user requests. The network communication between servers under the 5G environment can ensure the real-time transmission of data. The request-offload operation mode of the DBC approach may slow the real-time responses of user requests.

V. Related Work

In this section, we briefly overview several relevant existing works that significantly influenced our proposed approach.

A. Edge Computing

For the constrained computing ability and energy of mobile devices, resource poverty is a critical obstacle for intelligent mobile applications [1]. However, computation-incentive tasks

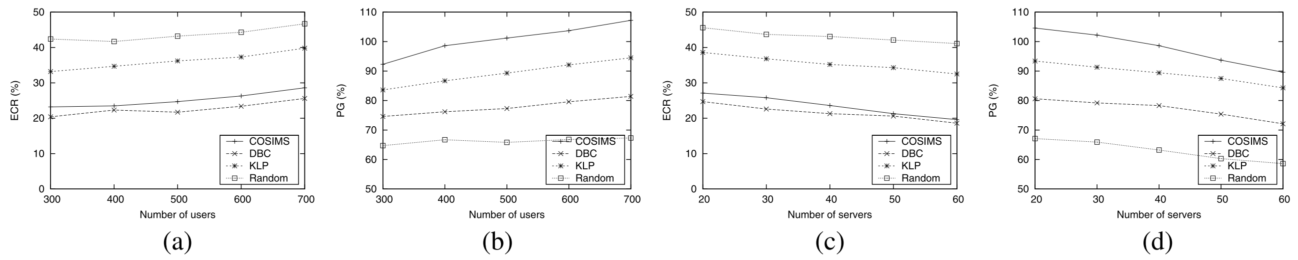

Fig. 5. Performance comparison for different approaches. (a) ECR (25 edge servers). (b) PG (25 edge servers). (c) ECR (600 mobile users). (d) PG (600 mobile users).

(e.g., image recognition, image rendering, etc.) in many emerging applications (e.g., VR/AR, autonomous driving, etc.) desire for a high real-time computation processing ability of the mobile devices [3], [4].

As recognized by the European 5G Infrastructure Public Private Partnership (5G PPP) research body, MEC is an emerging technology for 5G networks [14]. MEC aims at deploying edge services closer to the mobile devices to facilitate the backbone data traffic and response latency when some computation tasks in the mobile devices are desired to be offloaded to the remote center clouds in traditional cloud systems [34].

The cloudlet was first proposed to provide resource-rich edge computers or clusters of computers to rapidly instantiate customized service software and is available for use by nearby mobile devices [9]. Function as thin clients, mobile devices can properly process computing-intensive tasks by leveraging transiently customized proximate cloudlets. Ad-hoc cloud is another option that enables edge computing which directly performs the cloud computing in the combined several mobile devices [7]. Fog computing is a more general computing paradigm for performing edge computing, which utilizes the idle computation resources of billions of connected devices at the edge of network and enables the processes of the applications [35].

Besides cloudlets, micro DCs (or microclouds) can also be deployed around the world at user intensive areas to provide edge service to the mobile devices [12], [13]. In the 5G era, the enhanced mobile base stations can also provide some computing services [14]. Combining with cloud computing, base stations in the 5G environment can be cast as an edge server. In sum, we can represent MEC paradigm as a three-layer structure, i.e., user layer (i.e., mobile devices), edge layer (i.e., cloudlets, micro DCs, and base stations) and cloud layer (i.e., center cloud).

To guarantee the performance of cloud computing system, different fault tolerance approaches have been widely investigated. To identify the potential performance bottleneck in a composite cloud system, ARCMeas approach was proposed in [36]. A PSO-based cost-effective fault tolerance strategy was designed based on ARCMeas to guarantee the overall reliability as well as Quality of Service (QoS) of cloud systems.

Different from traditional QoS assurance researches for cloud systems, the resource gap of mobile devices is the main bottleneck of MEC applications. Computation offloading for mobile devices for MEC applications is a hot research topic in recent works [4].

B. Edge Service Computation Offloading

In sum, there are two different types of edge service offloading directions.

First, offloading from mobile devices to the edge servers. MobiCoRE [20] was proposed to enhance the cloudlet resources using mobile devices. Deng *et al.* [21] investigated the mobile service workflow's computation offloading problem. Based on genetic algorithm (GA), an optimal offloading strategy was proposed to concurrently minimize execution time and energy consumption for running composite services (i.e., workflows) in mobile devices. To offload the computing tasks of concurrent workflows in mobile devices to the cloudlets or center cloud, the multiobjective optimization offloading method, namely, COM, was proposed based on nondominated sorting GA III (NSGA-III) [16].

Chen *et al.* [17] studied the multihop computation offloading problem for different computing tasks under a blockchain-empowered Industrial IoT (IIoT) environment. A distributed game algorithm was proposed to minimize the computing cost in considering the distance and the task type (i.e., data processing and mining tasks) for IIoT devices by autonomously achieving the Nash equilibrium (NE).

To minimize the offloading cost of user tasks that can be offloaded over cloudlets, Mazouzi *et al.* [22] defined a mixed-binary programming problem by together considering tasks' execution time and mobile devices' energy consumption. Based on Lagrangian decomposition, a distributed linear relaxation-based heuristic approach was proposed to solve this problem.

Second, offloading computing tasks for edge services. To minimize the total energy consumption and response delay and guarantee the reliability of MEC system, cloudlets deployment problem in software-defined networking (SDN)-based IoT networks has been investigated [8], [18]. Yang *et al.* [18] defined this problem as a cloudlet placement and task allocation (CPTA) problem for network planning, and they studied how to place cloudlets on the network and allocate each requested task to cloudlets and public cloud with the minimum energy consumption. To balance the workload between cloudlets, Rodrigues *et al.* [19] proposed a PSO-based VM migration and transmission power control algorithm for scalable MEC.

C. Optimization Approaches

Edge service computation offloading problem is cast as a multiobjective optimization problem and many optimization techniques have become very popular recently. Abualigah [37] summarized the Optimization algorithms into four categories, which include local search-based algorithms, evolutionary search-based algorithms, swarm search-based algorithms, and hybrid algorithms.

Among them, evolutionary learning approaches in terms of metaheuristic, such as GA, ant colony optimization (ACO), PSO, gray wolf optimizer (GWO), and simulate anneal (SA), etc. are fairly well known. The applications of heuristic optimization are extensive [38]–[41]. Inspired by nature evolution process, most heuristic algorithms randomly update the parameters to approach the optimal solution by defining a fitness function. The heuristic algorithms can be easily used for many application scenarios. But there still lacks a strict theoretical foundation for heuristic algorithms. Too many parameters make it difficult to control the optimization effect in practical applications [42].

To solve the OT problem, Sinkhorn's matrix scaling algorithm is being increasingly used to unlock various problems, such as imaging sciences, computer vision and graphics, regression, classification, density fitting, etc. [24]. Comparing with the evolutionary algorithms, the main advantage of Sinkhorn optimization algorithm lies in the strict mathematical theory foundation, i.e., the probability theory. Hence, the application of Sinkhorn algorithm should be based on a well defined and designed situation. This may block the broadly application of Sinkhorn algorithm. However, this algorithm adopts less parameters and can achieve better optimization results. It also appears lower computational complexity (i.e., $O(d^3 \log d)$ for the worst case, where d represents the dimensions number of the matrix [25]).

VRs/ARs are computing-intensive latency-sensitive applications [43], [44]. Wearable Ad Hoc VR/AR devices cannot provide sufficient communication bandwidth and computing capability. The aforementioned two types of offloading directions are simultaneously desired for VR/AR applications. Moreover, different type of edge layer servers has different performance traits. Even for the same type of edge services, the edge servers may be different, e.g., different hardware devices of micro DCs [45]. Each edge service has different load capacity. Hence, methodologies for offloading the computation tasks among edge services which can guarantee the overall system performance are essentially desired for MEC applications.

Different from the relevant existing works, we model computation offloading among edge servers as an optimal transportation problem. Each edge server has an offloading objective by considering the loading ability, the overall computation load balancing, and the minimal extra communications hops. Sinkhorn's matrix scaling algorithm is employed to address the optimal transportation problem. Experimental evaluations verified the effectiveness and efficiency of the proposed approach. Solved by Sinkhorn's matrix scaling, the proposed approach has the advantage of more effective optimization effect and lower computational complexity than the competitive approaches. The proposed approach is still a centralized approach. This is the main disadvantage of our approach. We will investigate the decentralized edge service computation offloading approaches in the future to see if that will accrue any further benefits.

VI. Conclusion and Future Work

To deal with the limitation of low computation ability of mobile devices, the computational offloading among diverse neighboring edge servers is essentially needed for MEC applications. In this article, we propose COSIMS, a computation offloading approach for edge services based on Sinkhorn's matrix scaling algorithm for VR/AR MEC applications. COSIMS approach utilizes entropy regularization by defining a Lagrange multiplier to approach the optimal solution. The matrix scaling by the fixed-point iteration approach is used to solve the optimization problem. Experimental evaluations under real-world edge computing data indicate the effectiveness and low computation complexity of COSIMS.

We layout the following future research directions of computation offloading for edge servers. First, we will investigate the optimal transportation issue and develop new computation offloading approaches based on deep learning technologies. Second, the real-time reliability issue of the MEC application is also important. We will together consider the reliability and the computing load of edge servers when selecting the computation offloading objectives and develop new edge service computation offloading approaches. Third, distributed computation offloading approaches influenced by flooding working mode in traditional peer-to-peer technologies may also fit for solving the computation offloading issue for edge servers. We will also investigate the flooding computation offloading approaches.

Acknowledgment

The authors thank Dr. Zuling Kang at Cloudera, Inc. and Xiaohu Zhu at Safe AGI for their constructive suggestions.

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[2] W. Shi and S. Dustdar, "The promise of edge computing," *IEEE Comput.*, vol. 49, no. 5, pp. 78–81, May 2016.

[3] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1584–1607, Aug. 2019.

[4] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, "Edge cloud offloading algorithms: Issues, methods, and perspectives," *ACM Comput. Surveys*, vol. 52, no. 1, pp. 1–23, Feb. 2019.

[5] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[6] X. Xia, F. Chen, Q. He, J. C. Grundy, M. Abdelrazek, and H. Jin, "Cost-effective app data distribution in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 31–44, Jan. 2021.

[7] W. Zhang, Y. Wen, J. Wu, and H. Li, "Toward a unified elastic computing platform for smartphones with cloud support," *IEEE Netw.*, vol. 27, no. 5, pp. 34–40, Sep./Oct. 2013.

[8] L. Zhao, W. Sun, Y. Shi, and J. Liu, "Optimal placement of cloudlets for access delay minimization in SDN-based Internet of Things networks," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1334–1344, Apr. 2018.

[9] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct.–Dec. 2009.

[10] G. Cui, Q. He, F. Chen, H. Jin, and Y. Yang, "Trading off between user coverage and network robustness for edge server placement," *IEEE Trans. Cloud Comput.*, early access, Jul. 10, 2020, doi: 10.1109/TCC.2020.3008440.

[11] Q. He *et al.*, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 515–529, Mar. 2020.

[12] V. Bahl, "Emergence of micro datacenter (cloudlets/edges) for mobile computing," in *Proc. Microsoft Devices Netw. Summit*, 2015, pp. 1–61.

[13] C. Li, Y. Xue, J. Wang, W. Zhang, and T. Li, "Edge-oriented computing paradigms: A survey on architecture design and system management," *ACM Comput. Surveys*, vol. 51, no. 2, pp. 1–34, Apr. 2018.

[14] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," ETSI, Sophia Antipolis, France, White Paper, 2015.

[15] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin, "Online collaborative data caching in edge computing," *IEEE Trans. Parallel Distributed Syst.*, vol. 32, no. 2, pp. 281–294, Feb. 2021.

[16] X. Xu *et al.*, "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Gener. Comput. Syst.*, vol. 95, pp. 522–533, Jun. 2019.

[17] W. Chen *et al.*, "Cooperative and distributed computation offloading for blockchain-empowered industrial Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8433–8446, Oct. 2019.

[18] S. Yang, F. Li, M. Shen, X. Chen, X. Fu, and Y. Wang, "Cloudlet placement and task allocation in mobile edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5853–5863, Jun. 2019.

[19] T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato, and K. Temma, "Cloudlets activation scheme for scalable mobile edge computing with transmission power control and virtual machine migration," *IEEE Trans. Comput.*, vol. 67, no. 9, pp. 1287–1300, Sep. 2018.

[20] M. Whaiduzzaman, A. Naveed, and A. Gani, "MobiCoRE: Mobile device based cloudlet resource enhancement for optimal task response," *IEEE Trans. Services Comput.*, vol. 11, no. 1, pp. 144–154, Jan./Feb. 2016.

[21] S. Deng, L. Huang, J. Taheri, and A. Y. Zomaya, "Computation offloading for service workflow in mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3317–3329, Dec. 2015.

[22] H. Mazouzi, N. Achir, and K. Boussetta, "DM2-ECOP: An efficient computation offloading policy for multi-user multi-cloudlet mobile edge computing environment," *ACM Trans. Internet Technol.*, vol. 19, no. 2, p. 24, 2019.

[23] C. Villani, *Optimal Transport: Old and New*, vol. 338. Berlin, Germany: Springer, 2009.

[24] G. Peyré and M. Cuturi, *Computational Optimal Transport: With Applications to Data Science*. Norwell, MA, USA: Now, 2019.

[25] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2013, pp. 2292–2300.

[26] R. Sinkhorn, "Diagonal equivalence to matrices with prescribed row and column sums," *Amer. Math. Month.*, vol. 74, no. 4, pp. 402–405, 1967.

[27] R. Sinkhorn, "Diagonal equivalence to matrices with prescribed row and column sums. II," *Proc. Amer. Math. Soc.*, vol. 45, no. 2, pp. 195–198, 1974.

[28] S. Wang, J. Xu, N. Zhang, and Y. Liu, "A survey on service migration in mobile edge computing," *IEEE Access*, vol. 6, pp. 23511–23528, 2018.

[29] A. Machen, S. Wang, K. K. Leung, B. Ko, and T. Salonidis, "Live service migration in mobile edge clouds," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 140–147, Feb. 2018.

[30] P. Lai *et al.*, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Proc. Int. Conf. Service Orient. Comput.*, 2018, pp. 230–245.

[31] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Trans. Cloud Comput.*, vol. 5, no. 4, pp. 725–737, Oct.–Dec. 2015.

[32] S. Josilo, "Decentralized algorithms for resource allocation in mobile cloud computing systems," Ph.D. dissertation, School Elect. Eng. Comput. Sci., KTH Roy. Inst. Technol., Stockholm, Swedan, 2018.

[33] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng, "Multi-hop cooperative computation offloading for industrial IoT-edge-cloud computing environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 12, pp. 2759–2774, Dec. 2019.

[34] X. Xu *et al.*, "Multiobjective computation offloading for workflow management in cloudlet-based mobile cloud using NSGA-II," *Comput. Intell.*, vol. 35, no. 3, pp. 476–495, 2019.

[35] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st ed. Workshop Mobile Cloud Comput. (MCC)*, 2012, pp. 13–16.

[36] L. Wang, "Architecture-based reliability-sensitive criticality measure for fault-tolerance cloud applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 11, pp. 2408–2421, Nov. 2019.

[37] L. M. Abualigah, "Multi-verse optimizer algorithm: A comprehensive survey of its results, variants, and applications," *Neural Comput. Appl.*, vol. 32, pp. 12381–12401, May 2020.

[38] L. M. Abualigah and E. S. Hanandeh, "Applying genetic algorithms to information retrieval using vector space model," *Int. J. Comput. Sci. Eng. Appl.*, vol. 5, no. 1, pp. 19–28, 2015.

[39] L. M. Abualigah, A. T. Khader, and E. S. Hanandeh, "A new feature selection method to improve the document clustering using particle swarm optimization algorithm," *J. Comput. Sci.*, vol. 25, pp. 456–466, Mar. 2018.

[40] L. M. Abualigah, A. T. Khader, E. S. Hanandeh, and A. H. Gandomi, "A novel hybridization strategy for krill herd algorithm applied to clustering techniques," *Appl. Soft Comput.*, vol. 60, pp. 423–435, Nov. 2017.

[41] L. M. Abualigah, A. T. Khader, and E. S. Hanandeh, "Hybrid clustering analysis using improved krill herd algorithm," *Appl. Intell.*, vol. 48, no. 11, pp. 4047–4071, 2018.

[42] Z. Zhou, Y. Yu, and C. Qian, *Evolutionary Learning: Advances in Theories and Algorithms*. Singapore: Springer, 2019.

[43] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler, "Toward low-latency and ultra-reliable virtual reality," *IEEE Netw.*, vol. 32, no. 2, pp. 78–84, Mar./Apr. 2018.

[44] P. Ananthanarayanan *et al.*, "Real-time video analytics: The killer app for edge computing," *IEEE Comput.*, vol. 50, no. 10, pp. 58–67, Oct. 2017.

[45] X. Xu, C. He, Z. Xu, L. Qi, S. Wan, and M. Bhuiyan, "Joint optimization of offloading utility and privacy for edge computing enabled IoT," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2622–2629, Apr. 2020.

Lei Wang received the Ph.D. degree in computer science from Southeast University, Nanjing, China, in 2017.

He is an Associate Professor with Nanjing Forestry University, Nanjing. He visited the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong, in 2019. His publications have appeared in international journals and popular conferences, e.g., the IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON SERVICES COMPUTING, the *Journal of Parallel and Distributed Computing*, ICSOC, ICWS, and SCC. His research interests mainly include service computing, software reliability engineering, and data mining.

Yunqiu Zhang is currently pursuing the master's degree with Nanjing Forestry University, Nanjing, China.

Her research interests mainly include AI finance, edge service computation offloading, and concept drift-aware temporal cloud service APIs recommendation.

Shuhan Chen is currently pursuing the master's degree with Nanjing Forestry University, Nanjing, China.

Her research interests mainly include runtime reliability analytics for cloud systems and edge service computation offloading.