Mobility-Aware Offloading and Resource Allocation for Distributed Services Collaboration

Haowei Chen, Shuiguang Deng, *Senior Member, IEEE*, Hongze Zhu, Hailiang Zhao, Rong Jiang, Schahram Dustdar, *Fellow, IEEE*, and Albert Y. Zomaya, *Fellow, IEEE*

Abstract—In mobile edge computing (MEC) systems, mobile users (MUs) are capable of allocating local resources (CPU frequency and transmission power) and offloading tasks to edge servers in the vicinity in order to enhance their computation capabilities and reduce back-and-forth transmission over backhaul link. Nevertheless, mobile environment makes it hard to draw offloading and resource allocation decisions under dynamical wireless channel state and users' locations. In real life, social relationship is also provably a significant factor affecting integral performance in collaborative work, which results in MUs decisions strongly coupled and renders this problem further intractable. Most of previous works ignore the impact of inter-user dependency (or data dependency among IoT devices). To bridge this gap, we study the service collaboration with master-slave dependency among service chains of MUs and formulate this combinational optimization problem as a mixed integer non-linear programming (MINLP) problem. To this end, we derive the closed-form expression of resource allocation solution by convex optimization and transform it to integer linear programming (ILP) problem. Subsequently, we propose a distributed algorithm based on Markov approximation which has polynomial computation complexity. Experimental result on real-world dataset substantiates the usefulness and superiority of our scheme, in terms of reducing latency and energy consumption.

Index Terms—Mobile edge computing, task offloading, resource allocation, dependency, collaborative computing

1 Introduction

In recent years, network data traffic substantially proliferates with excessive MUs, e.g., smart phones and wearable equipments. At the same time, the development of infrastructure of Internet, including base station and cloud server cluster, couldn't keep up with the pace of ultra-low latency demand of networks [1]. In the interest of resolving this huge burden, MEC is proposed to avert the inevitable burden by pushing computation capabilities from core to the edge of networks [2]. It has emerged as a prominent paradigm to cope with the tremendous Internet data, where most of computation tasks are completed at edge servers in proximity through ratio access network (RAN), instead of the distant cloud [3]. Mobile network operators (MNOs) are allowed to deploy edge servers in strategic locations flexibly to guarantee quality of experience (QoE) of users. Without transferred to core network, MEC saves backhaul latency and energy cost of a high order of magnitude, and relieves the traffic load [4].

However, there are practical concerns about how to overcome the inherent nature of resources constraints. Conventional quality of service (QoS) based algorithms turn into obsolete to achieve globally optimal solution, due to the sophisticated network state caused by mobile terminals (MT) [5], [6]. Uninterrupted computing and transmission of user devices requires cost-effective methods to sustainably work, since the limited capacity of battery and scarce outdoor charging infrastructures pose impediments [7]. At last, ultra-low response latency is necessitated so that results could reach to delay-sensitive applications instantaneously, like augmented reality and autonomous vehicles [8], [9], [10]. It's expected to adjust CPU frequency and transmission power adaptively to cope with wireless channel emergency [11]. Therefore, for users, how to realize the trade-off between energy consumption and latency, so as to reduce response latency at the same time that energy consumption is stablized at a small enough level has been raised as one of the biggest issue [12].

Some existing works demonstate that social relationship could be utilized to draw strategies, since user's social attributes determine what roles they play in a group [13], [14]. This kind of interactions is referred to as inter-user dependency [15],

- *Haowei Chen, Hongze Zhu, and Hailiang Zhao are with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310012, China. E-mail: {haowei98, 22021097, hliangzhao}@zju.edu.cn.*
- *Shuiguang Deng is with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310012, China, and also with the Institute of Intelligence Applications, Yunnan University of Finance and Economics, Kunming 650000, China. E-mail: dengsg@zju.edu.cn.*
- *Rong Jiang is with the Institute of Intelligence Applications, Yunnan University of Finance and Economics, Kunming 650000, China. E-mail: jiang_rong@aliyun.com.*
- *Schahram Dustdar is with Distributed Systems Group, TU Wien, 1040 Vienna, Austria. E-mail: dustdar@dsg.tuwien.ac.at.*
- *Albert Y. Zomaya is with High Performance Computing & Networking, School of Computer Science, University of Sydney, Camperdown, NSW 2006, Australia. E-mail: albert.zomaya@sydney.edu.au.*

Manuscript received 27 Apr. 2021; revised 4 Jan. 2022; accepted 7 Jan. 2022. Date of publication 13 Jan. 2022; date of current version 24 Mar. 2022.
This work was supported in part by the Key Research Project of Zhejiang Province under Grant 2022C01145 and in part by the National Science Foundation of China under Grants U20A20173 and 62125206. The work of Schahram Dustdar was supported in part by the Zhejiang University Deqing Institute of Advanced Technology and Industrilization (ZDATI).
(Corresponding author: Shuiguang Deng.)
Recommended for acceptance by Q. Zheng.
Digital Object Identifier no. 10.1109/TPDS.2022.3142314

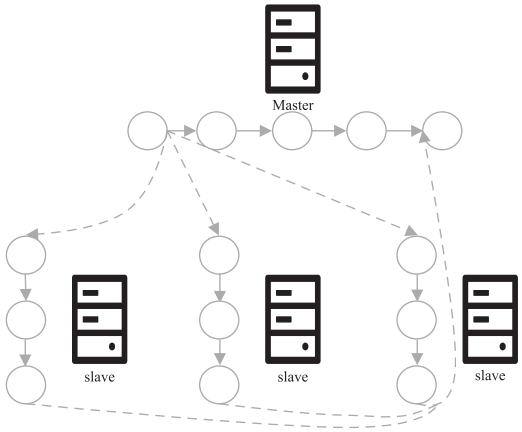


Fig. 1. Illustration of federated learning, where slaves are responsible for training models and master is required to send, update and aggregate parameters.

[16]. In addition, user mobility is also a useful feature for wireless network design [17]. However, few prior works combine both mobility and inter-user dependency into consideration.

Inter-user dependency is commonly appeared in real life as long as there exists groups (e.g., master-slave construction). People tend to experience web service with friends and families. It is more tractable to optimize overhead with regarding them as a whole. For the sake of elaboration, we use *service dependency* among individuals to abstract away from social attributes. It is common for team members to contact frequently and regularly to update schedule and require data from others according to what they need. In other words, service dependency correlates the service results of different users. For instance, only after the leader distributes work content to members, the latter could begin to work. It is ubiquitous in real-life business scenarios, such as Party A and B, superior and subordinate, and group work.

Note that this kind of service dependency could be found in technologies based on distributed system, especially master-slave-based architecture. Federated learning is a promising paradigm for training artificial intelligence (AI) models, where slave nodes train local model on private data samples solely and the master aggregate parameters (e.g., weights and biases of a deep neural network) from slave nodes and broadcasting them back for the next training round [17], as shown in Fig. 1. It's known that workload scheduling is the bottleneck of reducing latency. There exist some works aiming at utilizing MEC to improve the performance of federated learning. Yao *et al.* investigate the CPU frequency and transmission power controll of devices to optimize federated learning in MEC-aid network [18]. CSARO proposed in this paper is also a beneficial attempt to achieve acceleration at communication layer for federated learning in MEC network.

As for collaboration research for MEC system, Xia *et al.* studied collaborative data caching in edge servers with user dependency considered [19]. In other distributed systems, like wireless sensor network (WSN) and Internet of things (IoT) systems, this dependency could be reflected in data dependency between sink node (e.g., gateway) and sensor nodes [20], [21]. There exist other lots of master-slave-based systems, like Kubernetes [22] and Hadoop [23]. In other words, our work aims at proposing valuable insights in the sense of promoting the quality of experience of users and reducing overheads for these above technologies.

However, it is challenging to take both mobility and inter-user dependency into consideration. First, it requires a moderate trade-off between energy consumption and completion latency. Blind pursuit of ultra-low latency may lead to intolerable energy consumption, since transmission under deep fading costs higher energy consumption for minimum data unit (MDU) [24]. Second, offloading and allocation decisions among users are strongly mutual coupled. If the master demands results urgently, slaves have to shorten the latency at the expense of more energy consumption. Otherwise, it can follow an energy-effective method provided that the whole latency is slightly or even no deteriorated. Where there is dependency there is optimization space to realize system optimum. At last, the network state is time-varied. The next task would be started only after its all immediate predecessors have been completed. How to offload tasks and allocate resources optimally thus emerges as a global optimization problem, which leads to high computation complexity.

This paper aims at the combinational optimization of offloading and resources allocation to reduce collaborative cost. The basic idea is adapted from [15], [16]. However, the problem and the environment we focus on are not the same. They investigate the inter-user dependency optimization between two static users with only one server and extend it to a multi-user scenario. We care about the optimization for task model existing in master-slave systems, such as federated learning and other technologies mentioned before. It consists of multiple mobile users and multiple edge servers. Additional concerns for us are not only the difficulty of solving problems due to scale but also the spatio-temporal causality in a dynamic environment caused by mobility. The "one climb" policy designed by Yan *et al.* is not applicable any more since the task model changes and the number of edge servers increases. The main contributions of this paper can be summarized as follows:

- Our work provides a valuable scheme by jointly optimizing offloading and allocation policies to realize acceleration and green computing for master-slave-based scenarios in MEC environment, such as federated learning and real-life collaborative services.
- Taking the weighted sum of latency and energy consumption as objective, we devote efforts to solve the practical problem of ensuring QoS and sustainable computing for devices with both mobility and inter-user dependency considered. This work is state-of-the-art and challenging due to the high complexity in the design of solution with strong coupling caused by inter-user dependency.
- We formulate this problem as a non-convex MINLP problem which can only be solved by exhaustive search with exponential solution space. To this end, by assuming offloading decision is given, we obtain the allocation strategy for transmission power and CPU frequency with convex optimization. The MINLP problem thus is simplified to ILP problem. Next, we propose a distributed algorithm based on Markov approximation to obtain the near-optimal solution within polynomial time.

- Based on real-world dataset, we take two state-of-the-art algorithms and other baseline algorithms as benchmarks. Experimental results show our uperior performance and robustness as anticipated.

The rest of this paper is organized as follows. Section 2 summarizes related work and the system model is introduced in Section 3. We formulate the joint optimization problem and present the algorithm design in Section 4. Experimental results are provided in Sections 5 and 6 discusses the multi-dependency scenario and prove the feasibility of our algorithm and concludes this paper.

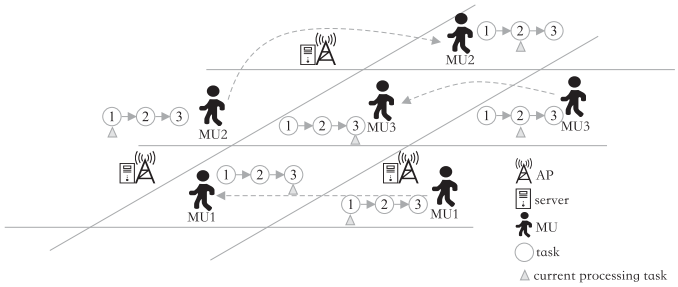


Fig. 2. MEC network with mobile users, APs and servers.

2 Related Work

Mobile edge computing technology emerges as a promising paradigm to alleviate the traffic load on core network with computation capability pushed to the edge of network. Due to the demands of QoS and limited battery capacity, there are a lot of prior works evolved to optimize latency and energy consumption in terms of resource allocation and offloading policy.

For offloading policy, previous literatures proposed many novel methods to solve optimization problem of multi-task offloading in MEC scenario. In [25], a game-theoretical approach was proposed with an objective to maximize number of served users and minimize overall latency and energy consumption by allocating users. In [26], a multi-user collaboration platform CoopEdge was developed which investigated reputation-based method for task executor selection where users with high reputations would be prioritized. In [27], based on Lyapunov optimization, Zhao *et al.* designed a general framework for offloading partitionable tasks to edge servers with the aim of minimizing response latency and stablizing battery of device in a reliable level. Qian *et al.* in [28] investigated NOMA for computation in multiaccess mobile edge computing and propose a deep reinforcement learning (DRL) method to solve the joint optimization problem of multi-task offloading, NOMA transmission and resource allocation to minimize the energy consumption of IoT devices. In [29], another DRL-based algorithm which can assimilate experiences and knowledge from the distributed system was studied by Qiu *et al.* to solve the multi-user computation offloading problem. With deep neuroevolution and policy gradient combined, the utilization of system was also improved. A minority game approach was proposed in [30] where users are modeled as players with the objective of addressing heterogeneous task offloading problem with incomplete information. In [31], a joint multi-hop multi-task partial computation offloading and network flow scheduling prolem was researched with the target of minimize the average delay of tasks in collaborative edge computing. By using McCormick envelope and Mosek solver in CVX, a lower bound solution was finally derived. The work [32] was oriented to computation offloading in industrial IoT-edge-cloud environment, aiming for QoS improvement. In [33], a privacy-preserving mechanism was proposed to make offloading decisions for DNN model training tasks and has been proved to reduce communication rounds.

Resource allocation is also an significant work in network optimization. The work [11] investigated the adjustable transmit power and CPU frequency problem in mobile environment and proposed an online algorithm to minimize users' response delay and energy consumption. In [34], Bahreini *et al.* studied the dynamic provisioning of computing resources. Two allocation and pricing mechanisms were proposed to maximize social welfare. [12] studied the energy allocation for users so as to locally execute computation tasks from energy beamforming. Chen *et al.* researched DVFS technology to realize energy efficient scheduling with adjustable CPU frequency [35]. Based on Lyapunov Optimization, they proposed a distributed algorithm which can achieve near optimal system profit while bounding the queue length. Li *et al.* developed an channel assignment and power control solution based on the idea of branch-and-price and then designed a greedy algorithm to achieve a sub-optimal performance [36]. Wu *et al.* investigated the optimal power allocation for NOMA relay-transmission and propose a layered-algorithm to efficiently compute the optimal power allocation solution and the maximum throughput for the targeted MU [37]. In [38], H-CRANs following online learning based centralized and decentralized approaches was proposed for transmission power control with inter-tier interference and capacity constraints.

As for the study of offloading and resource allocation for DAG, in [39], Meng *et al.* proposed an online deadline-aware offloading and scheduling algorithm called Dedas to determine both dynamical network bandwidth and computation task offloading with the objective of minimizing the average completion time. In [40], D-Dedas, a distributed algorithm was further designed for large network systems to allow edge servers making decisions independently while performance improved.

This paper aims to solve the joint optimization of offloading and resource allocation under dependency among users. In [16], Yan *et al.* establish inter-user dependency between two users and utilize "one-climb" policy to solve offloading and scheduling problem. After that, they carry on in-depth study in [15] with bi-section search method and Gibbs Sampling algorithm. However, the above studies only consider simple static scenario where exist one dependency, one edge server and two users. In this paper, we are devoted to resolve this problem in a multi-user, multi-server and multi-dependency mobile system where one-climb policy is inapplicable.

3 System Model

As illustrated in Figs. 2 and 3, we consider there is a project to be completed by a business sector, composed of multiple mobile users, denoted by $\mathcal{N} = \{1, 2, \ldots, N\}$. Each MU is required to finish a composition task chain. MU 1, as the master, is responsible for sending commands (generated from one

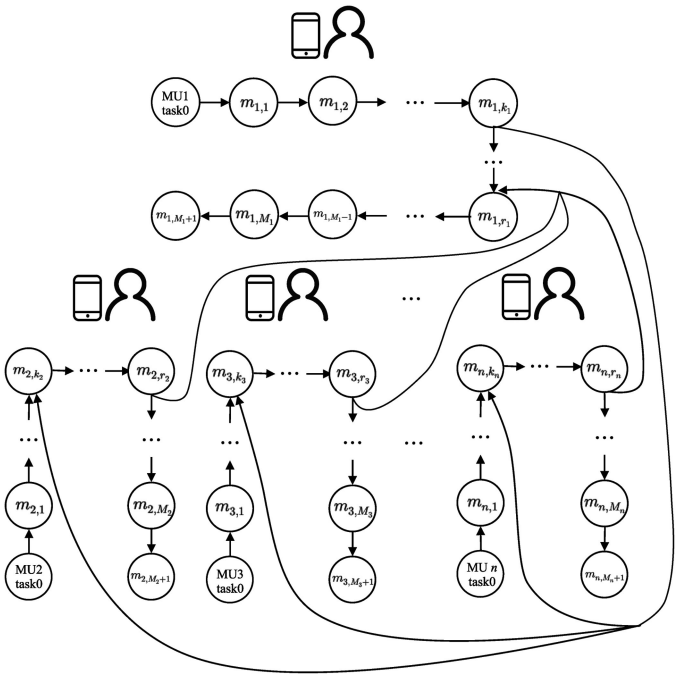


Fig. 3. Illustration of service dependency in task chains.

task) and aggregating results produced by other MUs to complete project (detailedly shown in Fig. 3). MUs $\{i|i \in \mathcal{N}, i \neq 1\}$ execute tasks individually and return demanded results back to MU1. It is exactly the same with federated learning that the central server orchestrates algorithms for training nodes (slaves) and aggregates newly updated parameters. The task model shown in Fig. 3 is a specific round of collaboration, like one training round in federated learning. Each task can be offloaded to edge servers or processed locally. Our optimization objective is to provide valuable schemes by drawing optimal offloading and allocation policies to realize acceleration and green computing for each round of collaboration for this kind of architecture, in mobile and dependent environment. We assume the master (MU 1) needs to interact with others twice, once to publish contents and once to collect results, i.e., broadcasting averaged gradients and aggregating updated gradients in federated learning. Key system parameters are shown in Table 1.

Task Model. For clarity, we show computation task chains of MUs detailedly in Fig. 3. Specifically, MU $i, \forall i \in \mathcal{N}$ has M_i computation tasks to be executed in a sequence with *inner-user* and *inter-user* dependency, defined as $\mathcal{M}_i = \{1, 2, \ldots, M_i\}$. We take $m_{i,j}$ to indicate jth task of $\mathcal{M}_i, \forall i \in \mathcal{N}, j \leqslant M_i$. Each task $m_{i,j}$ gets ready for execution only after outputs of its all predecessors have been received. For instance, m_{2,k_2} could be started after results of m_{1,k_1} and m_{2,k_2-1} have reached. We use $I_{i,j}$ and $O_{i,j}$ to indicate the input and output data of task $m_{i,j}$, respectively. Workload of task $m_{i,j}$ is represented by $D_{i,j}$. For convenience, we introduce two pseudo tasks for each user $i, \forall i \in \mathcal{N}$, referred as to $m_{i,0}$ and m_{i,M_i+1}, whose $I_{i,0} = O_{i,M_i+1} = 0$ and $D_{i,0} = D_{i,M_i+1} = 0$. We take DAG form $G = (V, E)$ to represent the whole project, where V indicates the set of tasks and E describes the dependency between two tasks. MU1 needs to distribute work contents to other MUs $\{i|i \in \mathcal{N}, i \neq 1\}$ and aggregate their results together, after $m_{1,k_1}, 1 \leq k_1 \leq M_1$ and m_{1,r_1}, respectively.

TABLE 1
Key System Parameters

Symbols	Definition
$\mathcal{S}, S$	Set of edge servers and the size of $\mathcal{S}$
$\mathcal{N}, N$	Set of mobile users and the size of $\mathcal{N}$
$\mathcal{M}_i, M_i$	Set of tasks for MU i and the size of $\mathcal{M}_i$
$m_{i,j}$	jth task for ith MU
$I_{i,j}, O_{i,j}$	Data size of input and output for task $m_{i,j}$
$D_{i,j}$	Workload of task $m_{i,j}$
$f_{i,j}$	CPU frequency of local execution for task $m_{i,j}$
$\tau^d_{i,j}$	Download delay for task $m_{i,j}$ if local execution
$TR^l_{i,j}$	Transmission delay of local execution for task $m_{i,j}$
$\tau^u_{i,j}$	Upload delay for task $m_{i,j}$ if edge computing
$E^u_{i,j}$	Upload energy consumption for task $m_{i,j}$
$TR^c_{i,j}$	Transmission delay of edge computing for task $m_{i,j}$
$CL^l_{i,j}$	Complete latency for task $m_{i,j}$ if local execution
$E^l_{i,j}$	Energy consumption for task $m_{i,j}$ if local execution
$R^s_{i,j}$	Bitrates for uploading task $m_{i,j}$
$R^c_{i,j}$	Bitrates for downloading task $m_{i,j}$
$p_{i,j}$	Transmitted power for task $m_{i,j}$
$h_{i,j}$	Channel gain when processing task $m_{i,j}$
$CL^c_{i,j}$	Complete latency for task $m_{i,j}$ if edge computing
$e^u_{i,j}$	Energy consumption for offloading task $m_{i,j}$
$A_{i,j}$	Execution platform of task $m_{i,j}$
$RT_{i,j}$	Ready time of task $m_{i,j}$
$FT_{i,j}$	Finish time of task $m_{i,j}$
$CL_{i,j}$	Complete latency for task $m_{i,j}$
$A_{i,j}$	Execution platform of task $m_{i,j}$
$\tau^{ex}_{i,j}(i',j')$	Extra latency between tasks $m_{i,j}$ and $m_{i',j'}$
$e^{ex}_{i,j}(i',j')$	Extra energy cost between tasks $m_{i,j}$ and $m_{i',j'}$
T_i	Overall latency to complete task chain $\mathcal{M}_i$
E_i	Overall energy cost to complete task chain $\mathcal{M}_i$
ω^T_i	weight parameter of latency for MU i
ω^E_i	weight parameter of energy consumption for MU i

Network Model. Edge servers are deployed by mobile network operator in strategic points to finish computation tasks on behalf of MUs, denoted by $\mathcal{S} = \{1, 2, \ldots, S\}$. Each server is equipped with an access point (AP) so as to realize communications via wireless LAN (WLAN). APs have different atenna allocation schemes and edge servers are heterogeneous in computation resources. MUs have options to offload tasks to edge servers or execute them locally. It's assumed that MUs can be only allowed to communicate with at most one edge server concurrently, which means offloading decision follows a binary policy [41]. We use $a^s_{i,j} \in \{0, 1\}$ to represent the binary decision of $m_{i,j}$, which follows

$$a^s_{i,j} = \begin{cases} 1, & \text{offloaded to edge server } s \\ 0, & \text{otherwise} \end{cases},$$

and

$$\sum_{s \in \mathcal{S}} a^s_{i,j} \leq 1. \tag{1}$$

Notice that pseudo tasks are naturally processed at local MUs, i.e., $\sum_{s \in \mathcal{S}} a^s_{i,0} = \sum_{s \in \mathcal{S}} a^s_{i,M_i+1} = 0$. Based on orthogonal frequency-division multiplexing (OFDM) technique, we consider an orthogonal partition on spectrum to enable multi-user offloading/downloading service. MUs communicate with edge servers over orthogonal channels and would not interfere with each other.

Mobility Model. We use Random Waypoint Model to characterizes MUs' mobility paths. Each user $i \in \mathcal{N}$ stays in the initial point $p_{i,0}$ for a random time $t_{i,0} \in (0, t_{\max}]$. Another point $p_{i,1}$ is next randomly selected. After stay, it moves to point $p_{i,1}$ at $v_{i,0}$ meters per second, where $v_{i,0}$ is randomly generated in $[v_{\min}, v_{\max}]$. Next, each user i stays in $p_{i,1}$ for a random time $t_{i,1} \in [t_{\min}, t_{\max}]$. This process is repeated until the user moves out. In this case, given an arbitrary time point $t \in (t_0, t_K)$, we can attain its location l_i. It's not difficult to verify that distance $d^s_{i,j}$ and channel gain $h^s_{i,j}$ are one-to-one mapping. Hence, there exists bijection between time point t and channel gain $h^s_{i,j}$.

3.1 Overheads of Local Execution

As the upgrade of central processing unit (CPU) and memory, the computation capability and storage have made a progress recently. Smart devices are enabled to process application locally, e.g., image preprocess and animation rendering [11].

Typically, local computing is an alternative strategy when channel state is poor. Dynamic voltage and frequency scaling (DVFS) technology provides the theoretical and technical support for dynamically adjusting the CPU frequency to execute tasks adaptively [42].

This subsection aims to analyse inner-user dependency and give the expression of energy consumption and latency brought by local execution for retrieving output of $m_{i,j-1}$ to complete $m_{i,j}$. Overheads caused by inter-user dependency will be discussed in Section 3.3. Task $m_{i,j}$ can be locally executed only after the output data of $m_{i,j-1}$ has reached the device of MU i. We define $TR_{i,j}$ to represent the transmission latency between tasks $m_{i,j-1}$ and $m_{i,j}$ as follows

$$TR^l_{i,j} = \sum_s a^s_{i,j-1}\left(1 - \sum_s a^s_{i,j}\right)\tau^d_{i,j}, \tag{2}$$

where $\tau^d_{i,j} = \frac{O_{i,j-1}}{R^c_{i,j}}$ is the download delay from edge server and $R^c_{i,j}$ is the downlink bitrate (explained in the next section). $TR^l_{i,j}$ only exists when task $m_{i,j-1}$ computed in edge server and task $m_{i,j}$ executed in local device, indicated by $\sum_s a^s_{i,j-1}(1 - \sum_s a^s_{i,j})$.

To quantify the completion latency, we let $f_{i,j}$ denote the CPU clock frequency for finishing task $m_{i,j}$. It thus can be expressed as

$$CL^l_{i,j} = \frac{D_{i,j}}{f_{i,j}}, \tag{3}$$

and the corresponding energy consumption of local execution is

$$E^l_{i,j} = (\alpha f^2_{i,j} + \beta)CL^l_{i,j} = \frac{\alpha D^2_{i,j}}{CL^l_{i,j}} + \beta CL^l_{i,j}, \tag{4}$$

where α and β are parameters decided by CPU models [43].

Notice that DVFS technology divides $f_{i,j}$ into discrete values ranging from 0 to f_{peak} actually. For simplicity, we regard $f_{i,j}$ as a continuous value to attain its optimal solution. Devices just need to choose the closest element from the set in practice.

3.2 Overheads of Edge Computing

In this subsection we will elaborate the concrete process of edge computing and how to calculate overheads brought by inner-user dependency. This process starts when task $m_{i,j-1}$ is finished and ends up with the completion of task $m_{i,j}$. It could be divided into two stages:

Offloading or Migrating. Before task $m_{i,j}$ to be edge computed, edge server should get the output of task $m_{i,j-1}$. Offloading happens when $m_{i,j-1}$ is completed locally, i.e., $\left(1 - \sum_s a^s_{i,j-1}\right)\sum_s a^s_{i,j} = 1$. MU i is required to send output data of task $j-1$ to target edge server with adjustable power $p_{i,j}$. It comes with transmission delay as

$$\tau^u_{i,j} = \frac{O_{i,j-1}}{R^s_{i,j}}, \tag{5}$$

where $R^s_{i,j}$ denotes the bitrate of uplink to edge server s which is quantified by

$$R^s_{i,j} = B\log\left(1 + \frac{p_{i,j}h^s_{i,j}}{\sigma^2}\right), \tag{6}$$

where B is the communication bandwidth, σ^2 denotes the variance of additive white Gaussian noise (AWGN) originating from receiver (e.g., receiver thermal noise) and $h^s_{i,j} = G\left(\frac{3\cdot 10^8}{4\pi F_c d^s_{i,j}}\right)^\theta$ denotes the channel gain caused by path loss and shadowing attenuation where G is the antenna gain, F_c indicates the carrier frequency, θ represents the path loss exponent and $d^s_{i,j}$ denotes the distance between target server s and MU i and θ represents the path loss exponent. $R^c_{i,j}$ is the downlink bitrate based on fixed power p^c.

By introducing $f(x) \triangleq \sigma^2(2^{x/B} - 1)$, (6) could be rewritten as

$$p_{i,j} = \frac{1}{h^s_{i,j}} f\left(\frac{O_{i,j-1}}{\tau^u_{i,j}}\right). \tag{7}$$

Accordingly, the energy overhead can be calculated by

$$E^u_{i,j} = p_{i,j}\tau^u_{i,j} = \frac{\tau^u_{i,j}}{h^s_{i,j}} f\left(\frac{O_{i,j-1}}{\tau^u_{i,j}}\right). \tag{8}$$

We take a triad $\{CPU_s, ST_s, BD_s\}$ to express the current status of each edge server $\forall s \in \mathcal{S}$ where CPU_s, ST_s, BD_s indicate the the percentage of remaining CPU, storage, and bandwidth resources where $0 \le CPU_s, ST_s, BD_s \le 1$. Task $m_{i,j}$ can be offloaded to edge server s only if its demand of these three resources would be satisfied, which can be expressed by $cpu_{i,j} \le CPU_s, st_{i,j} \le ST_s, bd_{i,j} \le BD_s$. Channel state is location-based since channel gain $h^s_{i,j}$ is related to the distance $d^s_{i,j}$ to edge server. It is thus essential for MUs to be aware of channel state information (CSI) before making strategies. [24] supposes APs are channel-aware while communicating and there exist feedback channels to transfer CSI from APs to MUs. Before uploading, MUs send offloading decisions to edge servers deployed with controllers. They determine how many resources $(cpu_{i,j}, st_{i,j}, bd_{i,j})$ the task will take up based on history records, check whether the offloading is feasible (sufficient computation capability,

bandwidth and storage resources) and send back control signal to MUs.

On the other hand, cross-edge migration happens when two consecutive tasks $m_{i,j-1}$ and $m_{i,j}$ are executed in two different edge servers. We use τ_0 to indicate the migration delay. Therefore, the latency for transmission is evaluate to

$$TR^c_{i,j} = \left(1 - \sum_s a^s_{i,j-1}\right) \sum_s a^s_{i,j} \tau^u_{i,j} + \sum_s a^s_{i,j-1} \left(1 - \sum_s a^s_{i,j-1} a^s_{i,j}\right) \tau_0. \quad (9)$$

The energy consumption of edge computing can be given by $E^c_{i,j} = \sum_s a^s_{i,j}(1 - \sum_s a^s_{i,j-1}) E^u_{i,j}$.

Edge Execution. Similar to local computing, the completion latency of edge computing can be given by

$$CL^c_{i,j} = \frac{D_{i,j}}{f^c_s}, \quad (10)$$

where f^c_s is the CPU frequency of sth edge server. Without loss of generality, we set $\min_{\forall s \in \mathcal{S}} f^c_s > f_{peak}$.

3.3 Overheads of Inter-User Dependency

The previous two subsections account for overheads resulted by inner-dependency at the same MU. In what follows, we will elaborate the impact of inter-user dependency on extra overheads, which can be quantified as the energy consumption and latency spent on fetching results from other MUs. For instance, task m_{2,k_2} needs to retrieve the output of m_{1,k_1}.

Let $A_{i,j}$ denote the execution platform of jth task in ith MU. $A_{i,j} = 0$ indicates task $m_{i,j}$ chooses local computing. Otherwise, $A_{i,j} = s$ means it is about to be offloaded to edge server s. Assume that $m_{i,j}$ is one predecessor of $m_{i',j'}$ where $i \neq i'$. Accoring to different $A_{i,j}$ and $A_{i',j'}$, we divide the extra overheads into five categories.

i) $0 < A_{i,j} = A_{i',j'}$

There is no need for extra communication since tasks $m_{i,j}$ and $m_{i',j'}$ can be executed consecutively at the same edge server.

ii) $0 = A_{i,j} < A_{i',j'}$

It means the result of computation task $m_{i,j}$ is required to be sent from MU i to $A_{i',j'}$. The additional overheads in terms of delay and energy consumption could be given by

$$\tau^{ex}_{i,j}(i',j') = \frac{O_{i,j}}{R^{A_{i',j'}}_{i,j}}, \quad (11)$$

and

$$e^{ex}_{i,j}(i',j') = \frac{O_{i,j}}{R^{A_{i',j'}}_{i,j} h^{A_{i',j'}}_{i,j}} f\left(\frac{O_{i,j}}{\tau^{ex}_{i,j}(i',j')}\right), \quad (12)$$

respectively. In this case, MU i dispatches result $O_{i,j}$ to $A_{i',j'}$ at bitrate $R^{A_{i',j'}}_{i,j}$ with a constant power p_0.

iii) $0 = A_{i',j'} < A_{i,j}$

Likewise, it can be expressed as an additional downloading from $A_{i,j}$. The extra cost can be computed by

$$\tau^{ex}_{i,j}(i',j') = \frac{O_{i,j}}{R^c_{i',j'}} \quad (13)$$

iv) $0 < A_{i,j} \neq A_{i',j'}$

In this case, $m_{i,j}$ and $m_{i',j'}$ are executed on different edge servers. As aforementioned before, the migration delay is $\tau^{ex}_{i,j} = \tau_0$.

v) $0 = A_{i,j} = A_{i',j'}$

This happens when the two tasks are executed in two local devices i and i', respectively. It needs to exchange result of $m_{i,j}$ from MU i to MU i' where APs act as relay nodes. The extra latency is

$$\tau^{ex}_{i,j}(i',j') = \frac{O_{i,j}}{R^{max}_{i,j}} + \frac{O_{i,j}}{R^c_{i',j'}} \quad (14)$$

where $R^{max}_{i,j}$ is the maximum uploading rate MU i could achieve with edge servers $\mathcal{S}$. It can be regared as once uploading and downloading process. Accordingly, the extra energy consumption is $e^{ex}_{i,j}(i',j') = \frac{p_0 O_{i,j}}{R^{max}_{i,j}}$.

3.4 Problem Formulation

To formulate the expression of delay T_i of each MU $i, \forall i \in \mathcal{N}$, we use the finish time FT_{i,M_i+1} of task m_{i,M_i+1} to equivalently represent the total latency of MU i. According to Section 3, we could express the finish time of the jth task of MU i, i.e., $m_{i,j}$, as a sum of two parts

$$FT_{i,j} = RT_{i,j} + CL_{i,j}, \quad (15)$$

where $RT_{i,j}$ is the ready time for $m_{i,j}$ to be executed and $CL_{i,j} = \sum_s a^s_{i,j} CL^c_{i,j} + (1 - \sum_s a^s_{i,j}) CL^l_{i,j}$ denotes the completion latency of task $m_{i,j}$. For any task, it is ready to be executed only after the outputs of its all predecessors have reached its execution platform. Let $pred(m_{i,j})$ indicate the predecessor task set of task $m_{i,j}$. $RT_{i,j}$ is given by

$$RT_{i,j} = \max\left\{FT_{i,j-1} + TR^l_{i,j} + TR^c_{i,j}, \mathcal{T}^{max}_{i,j}\right\}, \quad (16)$$

where in the $\max\{\cdot\}$ function, the former indicates the arrival time of the output of predecessor task from MU i, i.e., $m_{i,j-1}$, and the latter $\mathcal{T}^{max}_{i,j}$ denotes the latest arrival time of predecessors from other MUs $\{i' | i' \in \mathcal{N}, i' \neq i\}$, $\mathcal{T}^{max}_{i,j} \triangleq \max_{i' \neq i, m_{i',j'} \in pred(m_{i,j})} \left\{FT_{i',j'} + \tau^{ex}_{i',j'}(i,j)\right\}$. When $pred(m_{i,j})$ has only one element $m_{i,j-1}$, we set $\mathcal{T}^{max}_{i,j} = 0$, i.e., $RT_{i,j} = FT_{i,j-1} + TR^l_{i,j} + TR^c_{i,j}$. As a result, we can obtain the expressions of the overall latency T_i of MU i as

$$T_i = FT_{i,M_i+1}. \quad (17)$$

On the other hand, given any task $m_{i,j}$, we express its energy consumption as

$$E_{i,j} = \left(1 - \sum_s a^s_{i,j}\right) E^l_{ij} + \sum_s a^s_{i,j} E^c_{i,j} + e^{ex}_{i,j}. \quad (18)$$

The overall energy consumption of MU i can be evaluated to the sum of energy consumption of tasks from $m_{i,1}$ to $m_{i,j+1}$ expressed as

$$E_i = \sum_{i=1}^{M_i+1} E_{i,j}. \quad (19)$$

In this paper, our objective is to jointly optimize the energy consumption and response delay which are proved to be two competitive objectives hereinbefore in project collaboration. In real life, each participant in collaboration is supposed to be self-governed. They have their own concerns about the energy consumption and response delay to complete the whole project. For instance, there exists someone who is urgent to finish work (large ω_i^T), even at the cost of high energy consumption (small ω_i^E). Parameters ω_i^T, ω_i^E will influence other workers' decisions to reduce response time. In other words, what we care obout is not only the performance of the whole project, but also the concerns of each participant. We take the weighted sum of response delay and energy consumption of each participant minimization as our objective, defined as

$$\begin{aligned} P1: \min_{(a,p,f)} & \sum_{i\in N}\left(\omega_i^T T_i + \omega_i^E E_i\right) \\ \text{s.t.} \quad & 0 \le p_{i,j} \le p_{peak}, \\ & 0 \le f_{i,j} \le f_{peak}, \\ & a_{i,j}^s \in \{0,1\}, \forall i, j. \end{aligned} \tag{20}$$

where $0 < \omega_i^T < 1$ is the normalized weight parameter indicating how much MU i cares the whole latency, and normalized $0 < \omega_i^E < 1$ denotes the importance of energy consumption, which are decided by the preference of MU i and its specific application. It is interesting to mention that parameters ω_i^T and ω_i^E are allowed to take any value before normalization and when they are set to specific values, $P1$ can be interpreted as some special problems, such as max span minimization problem. Notice that solutions ($a_{i,j}^s, f_{i,j}$ and $p_{i,j}$) of tasks $m_{i,j}, \forall i \in \mathcal{N}, j \in \mathcal{M}_i$ are spatio-temporal coupled with each other. This kind of relations is reflected in the changes of network status brought by different locations of MU i, which is determined by the ready time $RT_{i,j}$ and mobility pattern. For inner-user dependency, the location where task $m_{i,j}$ is ready for execution is depended on how long previous tasks $m_{i,j'}, 1 \le j' \le j-1$ take, which depends on $a_{i,j'}^s, f_{i,j'}, p_{i,j'}$. For inter-user dependency, any task which needs results from other MUs cannot start execution unless results have been received. Once the location to execute task $m_{i,j}$ changes, the distance $d_{i,j}^s$ from MU i to edge server $s, \forall s \in \mathcal{S}$ will change accordingly and further affect the decisions and performance of later tasks. That is to say, solutions $a_{i',j}^s, f_{i',j}, p_{i',j}, \forall j \in \mathcal{M}_i'$ of other MUs also influence the performance of MU i. It is time-consuming to resolve this problem which is NP-hard. For simplicity, we assume resources of each edge server are enough for MUs.

4 Solution

This section outlines our solution of resource allocation and offloading, respectively.

4.1 Allocation of CPU Frequency and Power

We denote *receive* process as receiveing results from other MUs and *send* process as sending output to other MUs. As shown in Fig. 3, each MU has once receive process and once send process. It's not difficult to verify that only receiving output from others will influence subsequent tasks. To decouple inter-user dependency, we introduce N auxiliary variables $\mathcal{Q}_i, 1 \le i \le N$ to indicate the end time of receive process for MU i. MU 1 finishes receiveing after all results of other MUs have reached at its execution platform. $\mathcal{Q}_1$ can be given by

$$\mathcal{Q}_1 \triangleq \max\left\{FT_{1,r_1-1} + TR_{1,r_1}^l + TR_{1,r_1}^c, \mathcal{T}_{1,r_1}^{max}\right\}, i = 1. \tag{21}$$

As for MU $i, 1 < i \le N$, it only receives the result of m_{1,k_1} from MU 1. The $\mathcal{Q}_i$ can be given by

$$\mathcal{Q}_i \triangleq \max\left\{FT_{i,k_i-1} + TR_{i,k_i}^l + TR_{i,k_i}^c, \mathcal{T}_{i,k_i}^{max}\right\}, 1 < i \le N. \tag{22}$$

We could rewrite $RT_{i,j}$ as

$$RT_{i,j} = \begin{cases} \mathcal{Q}_1 & i = 1, j = r_1 \\ \mathcal{Q}_i & 1 < i \le N, j = k_i \\ FT_{i,j-1} + TR_{i,j}^l + TR_{i,j}^c & \text{otherwise} \end{cases}, \tag{23}$$

while the coupling term $\max\{\cdot\}$ is already eliminated.

Notice that there exists bijection between $f_{i,j}$ and $CL_{i,j}^l$, $p_{i,j}$ and $\tau_{i,j}^u$, which means we can obtain $f_{i,j}$ and $p_{i,j}$ by determining $CL_{i,j}^l$ and $\tau_{i,j}^u$, respectively. $P1$ could be further approximately reformulated as

$$\begin{aligned} & P1 - AP: \min_{(a,\tau^u,CL^l)} \sum_{i\in N}\left(\omega_i^T T_i + \omega_i^E E_i\right) \\ & \text{s.t.} \quad a_{i,j}^s \in \{0,1\}, \\ & 0 \le p_{i,j} \le p_{peak}, \\ & 0 \le f_{i,j} \le f_{peak}, \forall i, j, \\ & \sum_{i=2}^{N}\left(FT_{i,k_i-1} + TR_{i,k_i}^l + TR_{i,k_i}^c + \mathcal{T}_{i,k_i}^{sum} - 2\mathcal{Q}_i\right) \le 0, \quad \text{(C1)} \\ & FT_{1,r_1-1} + TR_{1,r_1}^l + TR_{1,r_1}^c + \mathcal{T}_{1,r_1}^{sum} - N\mathcal{Q}_1 \le 0, \quad \text{(C2)} \end{aligned} \tag{24}$$

where C1 and C2 approximate to (21) and (22), respectively, which indicate the inter-user constraints. Specifically, $\mathcal{T}_{i,j}^{sum} \triangleq \sum_{i'\neq i,(i',j')\in pred(i,j)} FT_{i',j'} + \tau_{i',j'}^{ex}(i,j)$. In this way, we decouple the solution of each MU. $P1 - AP$ is a non-convex MINLP problem which lacks of efficient algorithm. A closer observation of $P1 - AP$ shows that the feasible set of a is not related to τ^u, CL^l and vice versa. In other words, we can solve a and τ^u, CL^l separately. It is no difficult to verify that $P1 - AP$ would turn into a convex problem if a is given.

By assuming **a** is given, the Lagrangian function of $P1 - AP$ can be expressed as

$$\begin{aligned} L(\tau_{i,j}^l, \tau_{i,j}^u, \lambda, \mu) = & \sum_{i\in N}\left(\omega_i^T T_i + \omega_i^E E_i\right) \\ & + \lambda\sum_{i=2}^{N}\left(FT_{i,k_i-1} + TR_{i,k_i}^l + TR_{i,k_i}^c + \mathcal{T}_{i,k_i}^{sum} - 2\mathcal{Q}_i\right) \\ & + \mu\left(FT_{1,r_1-1} + TR_{1,r_1}^l + TR_{1,r_1}^c + \mathcal{T}_{1,r_1}^{sum} - N\mathcal{Q}_1\right), \end{aligned} \tag{25}$$

where λ and μ are Lagrange multipliers. Based on KKT condition, it is obtained that $\lambda \geq 0$ and $\mu \geq 0$.

Proposition 1 Adapted from Theorem 4.1 in [44] and Proposition 3.1 in [15]. *The optimal CPU frequency allocation $f^*_{i,j}$ of task j for MU i with $\sum_s a^s_{i,j} = 0$ is given by*

$$f^*_{i,j} = \min\left\{\sqrt{\frac{\mathcal{G}+\omega_i^E\beta}{\omega_i^E\alpha}}, f_{peak}\right\}. \tag{26}$$

For MU1, when $0 < j \leq k_1, \mathcal{G} = \mu + (N-1)\lambda$, when $k_1 < j \leq r_1 - 1, \mathcal{G} = \mu$, otherwise $\mathcal{G} = \omega_1^T$. For MU $i, 0 < i \leq N$, when $0 < j \leq k_i - 1, \mathcal{G} = \lambda$, when $k_i - 1 < j \leq r_i, \mathcal{G} = \omega_i^T + \mu$, otherwise $\mathcal{G} = \omega_1^T$.

Proof. For MU 1, when $0 < j \leq k_1$, the derivative of Lagrange function L with respect to $CL^l_{i,j}$ is

$$\begin{aligned}\frac{\partial L}{\partial CL^l_{1,j}} &= \frac{\partial\left(\omega_1^E E_1 + \mu F T_{1,r_1-1} + \lambda(N-1)FT_{1,k_1}\right)}{\partial CL^l_{1,j}} \\ &= \omega_1^E\left(-\frac{\alpha {D_{1,j}}^2}{{CL^l_{1,j}}^2} + \beta\right) + \mu + (N-1)\lambda,\end{aligned} \tag{27}$$

where $CL^l_{1,j}$ ranges within $\left[\frac{D_{1,j}}{f_{peak}}, +\infty\right)$ and $\frac{\partial L}{\partial CL^l_{1,j}}$ is a monotonously increasing function with $CL^l_{1,j}$. If $\frac{\partial L}{\partial CL^l_{1,j}}|_{CL^l_{1,j}=\frac{D_{1,j}}{f_{peak}}} > 0$, the optimal solution is $f^*_{1,j} = f_{peak}$. Otherwise, $f^*_{1,j} = \sqrt{\frac{\mu+(N-1)\lambda+\omega_1^E\beta}{\omega_1^E\alpha}}$. When $k_1 < j \leq r_1 - 1$, we have

$$\begin{aligned}\frac{\partial L}{\partial CL^l_{1,j}} &= \frac{\partial\left(\omega_1^E E_1 + \mu F T_{1,r_1-1}\right)}{\partial CL^l_{1,j}} \\ &= \omega_1^E\left(-\frac{\alpha {D_{1,j}}^2}{{CL^l_{1,j}}^2} + \beta\right) + \mu.\end{aligned} \tag{28}$$

Similarly, we can obtain $f^*_{i,j} = \min\left\{\sqrt{\frac{\mu+\omega_1^E\beta}{\omega_1^E\alpha}}, f_{peak}\right\}$. Otherwise, the derivative of L is

$$\begin{aligned}\frac{\partial L}{\partial CL^l_{1,j}} &= \frac{\partial\left(\omega_1^T T_1 + \omega_1^E E_1\right)}{\partial CL^l_{1,j}} \\ &= \omega_1^E\left(-\frac{\alpha {D_{1,j}}^2}{{CL^l_{1,j}}^2} + \beta\right) + \omega_1^T.\end{aligned} \tag{29}$$

Thus, we have $f^*_{i,j} = \min\left\{\sqrt{\frac{\omega_1^T+\omega_1^E\beta}{\omega_1^E\alpha}}, f_{peak}\right\}$. As for other MUs, the mathematical derivation of $f_{i,j}, 1 < i \leq N$ is similar with the proof of $f_{1,j}$. □

With mathematical analysis of Proposition 1, there are some interesting numerical properties. For the master, MU1, the optimal CPU frequency of $m_{1,j}, 0 < j \leq k_1$ is proportional to λ and μ (the larger λ and μ, the tighter constraints of inter-user dependency). When λ and μ exceed a certain threshold, MU1 has to compute tasks with the maximum frequency f_{peak} at the price of the maximum energy consumption. Likewise, as for $k_1 < j \leq r_1 - 1$, $f^*_{1,j}$ is only depended on μ, since task j has already been out of the impact range of the first dependency. Otherwise, when $r_1 - 1 < j \leq M_1$, $f^*_{1,j}$ is decided by ω_1^T/ω_1^E (i.e., comparison of preference on delay and energy consumption) instead of λ and μ. This is because the rest of $\mathcal{M}_1$ would not be affected by dependency, as independent individuals.

As for task $j, 0 < j \leq k_i - 1$ of MU $i, 1 < i \leq N$, $f^*_{i,j}$ is proportional to the value of λ. The larger λ means it is more urgent for ith MU to complete task k_i. As a result, MU i will execute tasks with larger CPU frequency regardless of more overhead. It is noted that $f^*_{i,j}$ for task $j, 0 < j \leq k_i - 1$ is unrelated with λ, since its performance has no direct impact on the second inter-dependency. On the other hand, $f^*_{i,j}$ for task $j, k_i - 1 < j \leq M_i$, is decided by λ. The rest tasks $j, r_i < j \leq M_i$ can be regarded as an independent task sequence. That is why their solutions $f^*_{i,j}$ are only depended on ω_i^T and ω_i^E.

Proposition 2. *The optimal transmit delay $(\tau^u_{i,j})^*$ of task $j, 1 \leq j \leq \mathcal{M}_i + 1$ for MU $i \in \mathcal{N}$ with $\left(1 - \sum_s a^s_{i,j-1}\right)\sum_s a^s_{i,j} = 1$ is given by*

$$(\tau^u_{i,j})^* = \begin{cases} \frac{O_{i,j-1}}{B\log_2(1+\frac{p_{peak}h^s_{i,j}}{\sigma^2})} & h^s_{i,j} \leq \frac{\sigma^2}{p_{peak}}[-\frac{z}{W(-ze^{-z})} - 1] \\ \frac{\ln 2 \cdot O_{i,j-1}}{B(W(ge^{-1})+1)} & \text{otherwise.} \end{cases} \tag{30}$$

For MU1, when $0 < j \leq k_1$, $z = 1 + \frac{\lambda(N-1)+\mu}{\omega_1^E p_{peak}}$ and $g = \frac{h_{1,j}(\lambda(N-1)+\mu)}{\omega_1^E\sigma^2} - 1$, when $k_1 + 1 < j \leq r_1 - 1$, $z = 1 + \frac{\mu}{\omega_1^E p_{peak}}$ and $g = \frac{h^s_{i,j}\mu}{\omega_1^E\sigma^2} - 1$, when $r_1 - 1 < j \leq M_1 + 1$, $z = 1 + \frac{\omega_i^T}{\omega_1^E p_{peak}}$ and $g = \frac{h_{1,j}\omega_1^T}{\omega_1^E\sigma^2} - 1$. For MU $\{i | i \in \mathcal{N}, i \neq 1\}$, when $0 < j \leq k_i - 1$, $z = 1 + \frac{\lambda}{\omega_i^E p_{peak}}$ and $g = \frac{h^s_{i,j}\lambda}{\omega_i^E\sigma^2} - 1$, when $k_i - 1 < j \leq r_i - 1$, $z = 1 + \frac{\mu}{\omega_i^E p_{peak}}$ and $g = \frac{h^s_{i,j}\mu}{\omega_i^E\sigma^2} - 1$, when $r_i - 1 < j \leq M_i + 1$, $z = 1 + \frac{\omega_i^T}{\omega_i^E p_{peak}}$ and $g = \frac{h^s_{i,j}\omega_i^T}{\omega_i^E\sigma^2} - 1$. $W(x)$ is Lambert function, the inverse function of $J(z) = zexp(z)$ [45].

Proof. It is similar with the proof for Proposition 1 and can be completed by deriving the the first-order and second-order derivative of Lagrangian function L with respect to $\tau^y_{i,j}$, where $i \in \mathcal{N}, j \in \mathcal{M}_i$. □

There are interesting properties that if channel gain $h^s_{i,j}$ is weaker than a certain threshold (e.g., caused by deep shading), MUs have to offload tasks with the maximum power as $p^*_{i,j} = p_{peak}$, since their results are demanded by other MUs with inter-user dependency. The threshold is inversely proportional to λ and μ, since tighter dependency means greater demand for $p^*_{i,j}$. Otherwise, when $h^s_{i,j}$ is higher than the threshold, MUs will obtain lower $(\tau^u_{i,j})^*$ since $W(x)$ is an increasing function when $x > -1/e$. In addition, like Proposition 1, λ, μ and ω^T do different affects according to the index of task.

Theorem 1. *We can fast obtain the optimal λ^* and μ^*, with iteration as*

$$\lambda(t+1) = [\lambda(t) + \ell(t)\cdot\mathcal{Y}]^+, \tag{31}$$

and

$$\mu(t+1) = [\mu(t) + \ell(t)\cdot\mathcal{Z}]^+, \tag{32}$$

where $\mathcal{Y} = \sum_{i=2}^{N} (2\mathcal{Q}_i - FT_{i,k_i-1} - TR^l_{i,k_i} - TR^c_{i,k_i} - \mathcal{T}^{sum}_{i,k_i})$ *and* $\mathcal{Z} = \left(N\mathcal{Q}_1 - FT_{1,r_1-1} - TR^l_{1,r_1} - TR^c_{1,r_1} - \tau^{sum}_{1,r_1}\right)$ *where* $\ell(t)$ *is diminishing stepsize [11].*

The number of iteration to converge of normal Lagrangian method is non-deterministic due to the random initial values of λ and μ. According to [11], [46], [47], if the diminishing stepsize satisfies $\sum_{n=0}^{\infty} \ell(n) = 0$ and $\lim_{t\to\infty} \ell(t) = 0$, Lagrangian method will fast converge, regardless of how awful the initial point is.

4.2 Solution of Offloading Policy

Hitherto, we obtain $p^*_{i,j}$ and $f^*_{i,j}$ by assuming $a_{i,j}$ is given. This MINLP problem is accordingly transformed into ILP program with variables $a^s_{i,j}$. It is a difficult problem since it is non-convex with exponential solution space as $\prod_{i\in\mathcal{N}} (M_i)^{S+1}$. In addition, due to the mobility of users and task dependencies, there exists spatio-temporal causality among offloading strategies.

The solutions of f^*, p^* hold the global optimum of the whole DAG, which makes it hard to determine the evolution equation between two states. In addition, the decision space is exponential. Thus dynamic programming is not an appropriate choice. Since $\mathcal{M}$ in MUs are heterogeneous and MUs are strongly coupled, centralized algorithm, e.g., evolution algorithm may not show good performance. Here, we introduce a distributed algorithm named CSRAO, based on Markov approximation to make it converge within polynomial convergence time [48], [49].

We define $\gamma_{i,j} \in \{0, 1, \ldots, S\}$ to indicate the offloading policy of task $m_{i,j}$, where $\gamma_{i,j} = 0$ represents local execution and $\gamma_{i,j} = s, 1 \le s \le S$ indicates offloading to server s. We denote Φ as the state set of Markov chain and $\phi \in \Phi$ as each state composed of offloading policy set $\{\gamma_1, \gamma_2, \ldots, \gamma_N\}$ where each γ_i is a M_i-dimension vector indicating the offloading decision of MU i and each dimension is $\gamma_{i,j}$ of task $m_{i,j}$. For brevity, we re-express $P1$ as a common optimization problem $\min_{\phi\in\Phi} x_\phi$, where $x_\phi = \sum_{i\in\mathcal{N}} \left(\omega^T_i T_i + \omega^E_i E_i\right)$.

To construct our problem-specific Markov chain, we associate each state with a percentage of time π_ϕ when ϕ is in use. Thus, it can be transferred to an equivalent minimum weight independent set (MWIS) problem, which hold the same optimal value, as

$$P1 - MWIS: \min \sum_{\phi\in\Phi} \pi_\phi x_\phi \quad s.t. \sum_{\phi\in\Phi} \pi_\phi = 1, \tag{33}$$

where we regard x_ϕ as the weight of π_ϕ. By introducing log-sum-exp function, $P1 - MWIS$ can be approximated to a convex problem

$$P1 - \rho: \min \sum_{\phi\in\Phi} \pi_\phi x_\phi + \frac{1}{\rho} \sum_{\phi\in\Phi} \pi_\phi \log \pi_\phi \quad s.t. \sum_{\phi\in\Phi} \pi_\phi = 1, \tag{34}$$

with approximation gap is upper-bounded by $\log|\Phi|/\rho$, where ρ is a positive constant. Based on KKT condition, the optimal value of (38) is given by $(1/\rho)\log\left[\sum_{\phi\in\Phi} \exp(-\rho x_\phi)\right]$ and the optimal solution is

$$\pi^*_\phi = \frac{\exp(-\rho x_\phi)}{\sum_{\phi'\in\Phi} \exp(-\rho x_{\phi'})}. \tag{35}$$

We consider π^*_ϕ as the stationary distribution to construct our time-sharing Markov chain. Once it converges, the time allocation π^*_ϕ for $\phi, \forall\phi \in \Phi$ can be obtained and $P1 - MWIS$ will be solved based on the most of time assigned to ϕ^* [49]. Due to the product form of π^*_ϕ, we can design at least one Markov chain, while satisfying

1) any two states are reachable from each other,
2) the following equation is satisfied

$$\pi^*_\phi q_{\phi,\phi'} = \pi^*_{\phi'} q_{\phi',\phi}, \tag{36}$$

where $q_{\phi,\phi'}$ denotes the transition rate from ϕ to ϕ'.

There exists many forms of $q_{\phi,\phi'}$. In this paper, we design it as follows:

$$q_{\phi,\phi'} = \frac{\exp(-\varphi)}{1 + \exp\left[-\rho\left(x_\phi - x_{\phi'}\right)\right]}, \tag{37}$$

where φ is a constant while the above two conditions are satisfied.

Algorithm 1. Collaborative Service Resource Allocation and Offloading (CSRAO) Algorithm

Input: Mobility information of each MUs, network state information and DAG information
Output: Resource allocation $p_{i,j}$ and $f_{i,j}$ and offloading policy $a^s_{i,j}$

1 **for** *each MU* $i \in \mathcal{N}$ **do**
2 Initialize the offloading policy γ_i randomly ;
3 Generate an exponential random timer with mean as $\exp(\varphi)/V_i$;
4 **end**
5 Calculate the objective $P1$ as λ and μ converge;
6 Denote the current system state as ϕ and let all MUs begin counting down;
7 **repeat**
8 **if** *there exists one timer of MU expires* **then**
9 Denote the MU as i and update its offloading policy randomly as new state ϕ' with only one element changed;
10 MU i broadcasts the new state to other MUs to obtain the value of f^* and p^*;
11 $\phi \leftarrow \phi'$ with probability $1 - \exp(-\rho x_\phi)/\left(\exp(-\rho x_\phi) + \exp(-\rho x_{\phi'})\right)$;
12 All MUs refresh timers and begin counting down;
13 **end**
14 **until** $P1$ *converges*
15 **return** γ, f^* and p^*;

We propose distributed CSRAO algorithm as shown in Algorithm 1, where MUs are just required to broadcast its new state with each other. At the beginning, all MUs initialize offloading policies and generate their own timers randomly (line 1-line 4). Accordingly, each MU i determines $\mathbf{f_i}$ and $\mathbf{p_i}$ from (26) and (30) (line 5, line 6). In each iteration, the first expiring MU randomly updates its offloading decision with only one element changed and informs other MUs to obtain new $\mathbf{f}$ and $\mathbf{p}$. System has to decide whether to stay in the new state according to (37) (line 8-line 13) until $P1$ converges. It's worth mentioning that CSRAO not only could

converges fast but also possesses the mechanism to prevent falling into local optimum with adjustable parameter ρ.

Complexity Analysis. Recall that we apply Lagrangian multiplier method to derive the closed-form expression of $f_{i,j}^*$ and $p_{i,j}^*$, by assuming a is given. In Algorithm 1, while updating $a_{i,j}$, $f_{i,j}^*$ and $p_{i,j}^*$ under diminishing stepsize, it could converge fast with complexity as $O(TM_i)$, where T represents the number of iteration. Due to $\ell(t)$, the value of T is small. The searching space of a is $O(\prod_{i\in N}(M_i)^{S+1})$. By introducing Markov approximation algorithm, we could obtain the sub-optimal solution within polynomial convergence time when ρ is a proper value in Theorem 4, which is significantly reduced from exponential complexity of exhaustive search.

4.3 Mechanism Analysis

In this subsection, we will analysis convergence feasibility, approximation gap and convergence time of our proposed CSRAO algorithm.

Theorem 2. *$P1$ can be solved globally by Algorithm 1, with transition probability as $\exp(-\rho x'_\phi)/\big(\exp(-\rho x_\phi)+\exp(-\rho x_{\phi'})\big)$. When $\rho \to 0$, Algorithm 1 converges with probability 1.*

Proof. Let ϕ denote the current state with a, f, p given. The objective value x_ϕ could be derived as (24). In each iteration, user i who expires first randomly change its offloading policy $a_{i,j}, \forall j \in \mathcal{M}_i$ to update a. In this case, f, p could converge fast according to (31) and (32), so as to obtain x'_ϕ. According to Algorithm 1, under state ϕ, user i counts down with rate as $V/\exp(\varphi)$, there is

$$q_{\phi,\phi'} = \frac{1}{V_i}\cdot\frac{\exp(-\rho x_\phi)}{\exp(-\rho x_{\phi'})+\exp(-\rho x_\phi)}\cdot\frac{V_i}{\exp(\varphi)} = \frac{\exp(-\varphi)}{1+\exp\big(-\rho\big(x_\phi - x_{\phi'}\big)\big)}, \tag{38}$$

which is equal with (37) designed before. It means our design is satisfied to realize time-sharing among different states in Markov chain. Thus, the optimal solution will be gained with stationary distribution converge to (35).

Let ϕ^* be the global optimum, i.e., $x_{\phi^*} \le x_\phi, \forall \phi \in \Phi$. According to (37), the system is prone to stay in x_ϕ with larger probability $q_{\phi,\phi'}$ whose objective value is lower. It indicates the system will converge to ϕ^* and be allocated with time percentage as π_ϕ^*. We re-express (35) as

$$\pi_{\phi^*}^* = \frac{1}{\sum_{\phi'\in\Phi}\exp(-\rho(x_{\phi^*}-x_{\phi'}))}. \tag{39}$$

It can be concluded that $\pi_{\phi^*}^*(x)$ increases as ρ decreases. When $\rho \to 0$, the time percentage allocated to ϕ^* is $\pi_{\phi^*}^* \to 1$. However, there exists a tradeoff that large ρ may lead to system getting trapped in local optimum. □

Theorem 3. *The approximation gap is upper-bounded by $0 \le \epsilon - \varepsilon \le \log|\Phi|/\rho$, where ϵ denotes the approximation solution of CSRAO and ε indicates the optimum [46], [50].*

Proof. Following [50], we introduce Dirac delta function to represent the distribution $\bar{\pi}_\phi$ of the theoretical optimality with definition $\phi_{min} \triangleq \arg\min_{\phi\in\Phi} x_\phi$.

$$\bar{\pi}_\phi = \begin{cases} 1, & \text{if } \phi = \phi_{\min} \\ 0, & \text{otherwise} \end{cases}. \tag{40}$$

According to the optimal stationary distribution of π_ϕ^*, we derive

$$\sum_{\phi\in\Phi}\pi_\phi^* x_\phi + \frac{1}{\rho}\sum_{\phi\in\Phi}\pi_\phi^*\log\pi_\phi^* \le \sum_{\phi\in\Phi}\bar{\pi}_\phi x_\phi + \frac{1}{\rho}\sum_{\phi\in\Phi}\bar{\pi}_\phi\log\bar{\pi}_\phi = \varepsilon. \tag{41}$$

Based on Jensen inequality, we obtain

$$\sum_{\phi\in\Phi}\pi_\phi^*\log\pi_\phi^* \ge -\log\left(\sum_{\phi\in\Phi}\pi_\phi^* * \cdot\frac{1}{\pi_\phi^*}\right) = -\log|\Phi|. \tag{42}$$

By substituting (42) to (41), we have

$$\epsilon = \sum_{\phi\in\Phi}\pi_\phi^* x_\phi \le \varepsilon + \frac{\log|\Phi|}{\rho}. \tag{43}$$

With $\epsilon > \varepsilon$, thus, Theorem 2 is proved. □

Theorem 4 Adapted from Theorem 5 in [47]. *The mixing time (convergence time) of our designed continuous-time Markov chain is upper-bounded as $O(\log(N))$ and lower-bounded if $\ln\frac{1}{2\nu}/2\exp(-\varphi-\rho x_{\min})\sum_{i\in N}\big(M_i{}^{S+1}-1\big), 0 < \rho < \rho_{th}$, where $\rho_{th} = \ln\left(1+\frac{1}{\sum_{i\in N}(M_i{}^{S+1}-1)}\right)/2(x_{\max}-x_{\min})$.*

Proof. Let $P_t(\phi)$ represent the probability distribution of all states in Φ at time t, with initial state as ϕ. According to [47], the mixing time can be given by

$$t_{mix}(\nu) \triangleq \inf\left\{t > 0 : \max_{\phi\in\Phi}\|P_t(\phi)-\pi^*\|_{TV} \le \nu\right\}. \tag{44}$$

We further denote $x_{\max} \triangleq \max_{\phi\in\Phi} x_\phi$ and $x_{\min} \triangleq \min_{\phi\in\Phi} x_\phi$. Since $|\Phi|\exp(-\beta x_{\max}) \le \sum_{\phi'\in\Phi}\exp(-\beta x_{\phi'}) \le |\Phi|\exp(-\beta x_{\min})$. According to (35), we could derive the minimum probability of stationary distribution $\pi_{\min} \triangleq \min_{\phi\in\Phi}\pi_\phi^* \ge \frac{\exp(-\rho(x_{\max}-x_{\min}))}{|\Phi|}$.

Based on uniformization technique [51], let $Q = \{q_{\phi,\phi'}\}$ denote the transition rate matrix of our Markov chain and develop a discrete-time Markov chain ς with transition rate matrix $P = 1 + \frac{Q}{\theta}$, where $\theta \triangleq \sum_{i\in N}\big(M_i{}^{S+1}-1\big)\exp(-\varphi-\rho x_{\min})$ is the uniformization constant and I is the identity matrix.

Applying spectral gap inequality [52], we have $t_{mix}(\nu) \ge \frac{1}{\theta(1-\varrho_2)}\ln\frac{1}{2\nu}$ where ϱ_2 indicates the second largest eigenvalue of P. With Cheeger's inequality [52], we have $1-2\chi \le \varrho_2 \le 1-\frac{1}{2}\chi^2$ where χ is bounded as $\frac{\pi_{\min}}{\theta}\cdot\exp(-\varphi-\rho x_{\max}) \le \chi \le 1$. $t_{mix}(\nu)$ satisfies

$$t_{mix}(\nu) \ge \frac{\ln\frac{1}{2\nu}}{2\exp(-\varphi-\rho x_{\min})\sum_{i\in N}\big(M_i{}^{S+1}-1\big)}. \tag{45}$$

Next, we prove the upper bound of $t_{mix}(\nu)$. Similarly, we construct a uniformized Markov chain ς' whose transition matrix is $P' = I + Q/\theta'$ by uniformization technique on our Markov chain. θ' is given by $\theta' \triangleq \exp(-\varphi)\sum_{i\in N}\big(M_i{}^{S+1}-1\big)\sum_{\phi\in\Phi}\exp(-\rho x_\phi)$. By path coupling method [53], there is

$$d_{TV}(P_t(\phi), p^*) \leq N \cdot \exp(-\exp(-\varphi)Kt\exp(-\rho(2x_{\max} - x_{\min}))), \quad (46)$$

where $K = \sum_{i\in N}\left[(M_i{}^{S+1} - 1)(\exp(2\rho(x_{\max} - x_{\min}))\right], 0 < \rho < \rho_{th} = \ln\left(1 + \frac{1}{\sum_{i\in N}(M_i{}^{S+1}-1)}\right)/2(x_{\max} - x_{\min})$. Thus, according to [47], we have

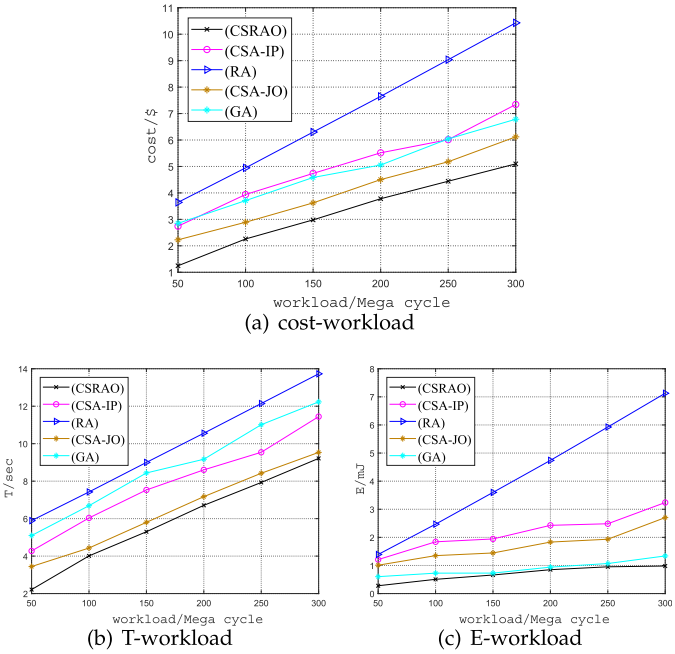
$$t_{mix}(\nu) \leq \frac{\exp(\rho(2x_{\max} - x_{\min}) + \varphi) \cdot \ln\frac{N}{\nu}}{\sum_{i\in N}\left[(M_i{}^{S+1} - 1)(\exp(2\rho(x_{\max} - x_{\min})))\right]}. \quad (47)$$

□

5 Experimental Result

Based on Random waypoint model, we simulate trajectories of MUs in a grid of 1000m × 1000m. $t_{\max}$ is set to 30s and the speed v of mobile user follows a uniform distribution in [4m/s, 15m/s]. Moreover, we exploit EUA dataset [54] to construct the distribution of 10 edge servers with randomly sampling them in our grid. With referring to [11], we assume there exists mobile users with SAMSUNG Galaxy S5 whose maximum CPU frequency is 2.5 GHz, and Dell I3650-1838 as edge server with quad-core 2.7 GHz CPU. As for each task $m_{i,j}$, the workload $D_{i,j}$, input $I_{i,j}$ and output data $O_{i,j}$ are assumed to follow a uniform distribution in [50,100] Mega cycles, [10,50] KB, and [10,50] KB, respectively. Channel bandwidth is set to $B = 5MHz$ with AWGN $\sigma^2 = -90dBm$ and date rate of downlink is $R_c = 100Mbps$. The peak transmit power is equal to $p_{peak} = 0.1$. As parameters of free-space path loss model in [15], We set path loss exponent $\theta = 3$, antenna gain $G = 4.11$, and carrier frequency $F_c = 915MHz$. For simplicity, we assume CPU, storage and storage resources of edge servers are large enough for collaboration tasks. As for local computation, α and β are set as 0.34 and 0.35 [55]. We initialize weight parameters ω_i^T and ω_i^E as 0.5 and 0.5, respectively. We consider the following benchmark methods for performance comparison:

- CSA-IP [15]: current state-of-the-art algorithm to solve inter-dependency problem. For offloading policy, it uses Gibbs sampling algorithm to obtain it iteratively. In addition, offloading decision for each user follows "one-climb" policy that tasks are either offloaded to the edge server for exactly once. Furthermore, it adopts Lagrange multiplier method to derive the closed form expression of CPU frequency and transmitted power.
- GA: genetic algorithm. Since genetic algorithm can only solve the integer programming problem, we reserve our Lagrange multiplier method to obtain the optimal solution of $f_{i,j}$ and $p_{i,j}$. Total offloading decision solution is regarded as a chromosome where gene is an offloading decision for each task. In each iteration, we execute crossover and mutate to generate new chromosomes and use Lagrange multiplier method to solve $f_{i,j}$ and $p_{i,j}$.
- CSA-JO [11]: current state-of-the-art algorithm to solve the joint optimization problem of resource allocation and offloading where mobile users are served by multiple servers without inter-user dependency. It utilizes Lagrangian multiplier method to obtain the optimal CPU frequency and transmission power, and selects edge server by setting a contention period for edge servers to compete for tasks.
- RA: Random algorithm. MUs randomly draw offloading decisions among all edge servers and local execution. Next, adjust CPU frequency or transmission power randomly.



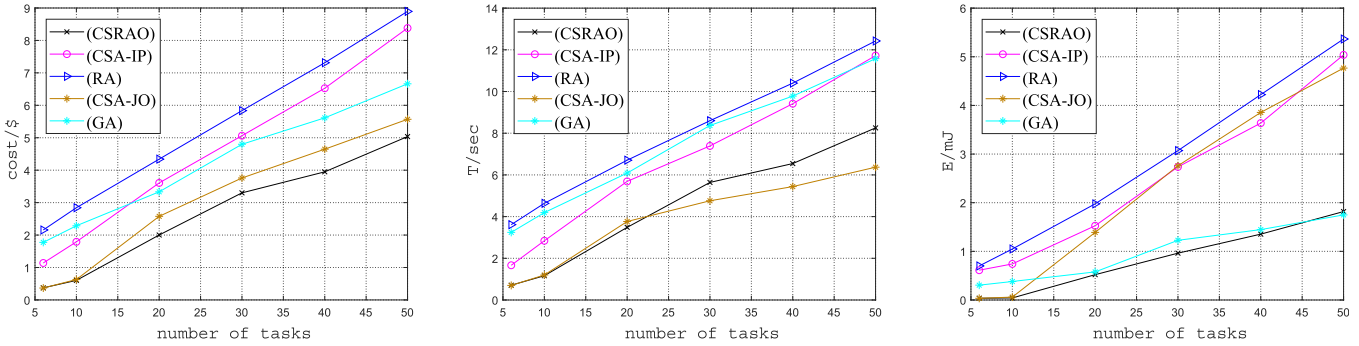
(a) cost-workload (b) T-workload (c) E-workload

Fig. 4. Comparison under increasing workload in terms of cost, response latency and energy consumption.

In what follows, we will compare our CSRAO algorithm with these four baseline algorithms in three aspects, feasibility, scalability and sensitivity, to verify our superiority.

5.1 Feasibility Validation Under Varing Workload

In this subsection, we compare performance of these four algorithms with workload $D_{i,j}$ varied from 50 to 300. We can observe from Fig. 4a that GA, CSA-IP and CSA-JO outperform RA in term of the cost for service collaboration, i.e., the weighted sum of latency and energy consumption. In addition, the solution of offloading and allocation policy in the case of dynamic task workload obtained by our algorithm is more cost-effective. When workload increases to 300 from 50, CSRAO is always the best scheme while the second is CSA-JO and the third is GA. That is because "one-climb" policy is not applicable to send-and-receive dependency in multi-user multi-server scenario. It is difficult for CSA-IP to find a good solution of CPU frequency and transmitted power. Meanwhile, although CSA-JO proposes a scheme to select one "best" server from candidates, it regards each MU as an individual node and accordingly neglects service dependency. As for GA, it is hard to get the optimal solution without consider the problem-specific factor due to the large decision space. On the contrary, our algorithm aims to jointly optimize offloading, CPU frequency and transmission power to achieve the global optimality.



(a) cost-number of tasks (b) T-number of tasks (c) E-number of tasks

Fig. 5. Comparison under increasing number of tasks in terms of cost, response latency and energy consumption.

To gain detailed performance of each scheme, we draw the variation tendency of two competing objectives, latency and energy consumption in Figs. 4b and 4c, respectively. Findings can be drawn including that (i) as workload increases, results of all methods increases; (ii) our scheme could achieve superior results in terms of both latency and energy consumption; (iii) even if GA spends the lowest energy only second to our CSRAO, its latency is too long to tolerant, which leads to its total cost to stay in a high level and worse than CSA-JO. The reason for first finding is larger workload naturally requires longer time to complete. The reason behind (ii) is our scheme considers inter-user dependency and can be applied to a more complex scenario. The last finding reveals that only stablizing one objective to a low level could not ensures the optimality of global cost.

5.2 Scalability Analysis

We take the four schemes into comparison in terms of system cost, response latency and energy consumption, with the number of users and tasks increasing.

5.2.1 Comparison With Increasing Number of Tasks

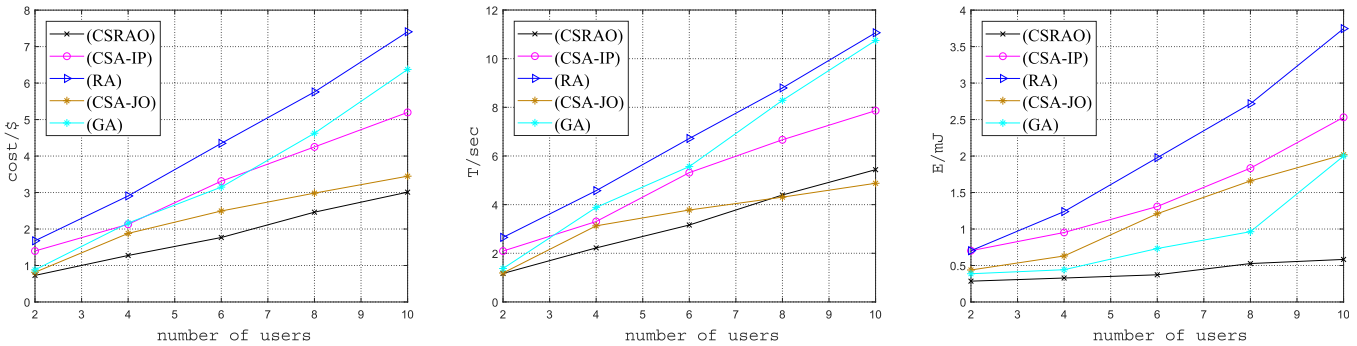
In this subsection, we evaluate the performance of our framework under different number of tasks, compared with other baseline schemes. As shown in Fig. 5a, we vary task number of each MUs M_i from 5 to 50. We can observe that CSA-JO and ours obtain close relatively optimal results and outperform others greatly while task number is 5. That is because when the number of tasks is small, the effect caused by inter-user dependency is also small. As tasks increase, the gap between ours and CSA-JO gets larger. In addition, our algorithm shows superior performance and remains the greatest level from beginning to end. When tasks exceed a certain threshold, GA scheme outperforms CSA-IP, due to mutation and crossover for global optimum. Our scheme facilitates MUs to reduce long-term energy and latency cost under location-based channel state and prevents being trapped in local optimum.

Next, we examine the performance tradeoff between the two competing objective in $P1$, completion latancy $\sum_{i\in N} T_i$ and energy consumption $\sum_{i\in N} E_i$. Figs. 5b and 5c reflect the impact of task number on latancy and energy consumption, respectively. Both of them show that varying task number from 5 to 50 naturally results in an increase of E and T for all methods. We can observe from Figs. 5b and 5c that RA has the poorest performance no matter whether it is latency or energy consumption. From Fig. 5b, CSA-JO acts very close to us and even outperforms ours later. However, we find it obtains too large energy consumption from Fig. 5c. As a result, CSA-JO ranks second to us in term of the weighted sum of latency and energy consumption. GA is also the same and CSA-IP has the poorest results. The reason behind this is CSA-IP requires a specific scenario that "one-climb" policy is tenable. We can come to a conclusion from Figs. 5a, 5b and 5c that even though there exists one algorithm could obtain one lower objective than ours, latency or energy consumption, our algorithm could realize better tradeoff. This finding is because as task number increases, more tasks are affected by others which means before making decisions, each MU has to take the performance of whole system into consideration, even at the expense of its private optimality. For instance, MU i has to speed up for transmission in order to improve social optimality even if deep loss happens and vice versa.

5.2.2 Comparison With Increasing Number of Users

In this subsection, we evaluate the influence of the number of users N on the optimality of objectives, including total cost, latency and energy consumption. It is not difficult to find our algorithm shows the best performance as the number of users increases from Fig. 6a. Since CSA-IP only considers two-user scenario, it shows poorness of scalability. At beginning, when user number is 2, the gap among results of GA, CSA-JO and ours is very small. As user number varies from 2 to 10, GA and CSA-JO behave powerless. Reasons behind it are that CSA-JO ignores the service dependency among MUs and it hard for GA to converges with an exponential decision space. Meanwhile, our scheme maintains the best performance all along. That means our CSRAO could achieve the tradeoff between MU itself and system optimality well.

We draw the results of two competitive objectives, latency and energy consumption in Figs. 6b and 6c. As user number increases, service dependency among MUs turns into more complicated. The tradeoff between latency and energy consumption becomes harder to realize. That means schemes have to draw offloading and allocation decisions under a more complex DAG. It's not difficult to find that the two subproblem are strongly coupled. Offloading policy determines the environment of allocation and the performance of offloading is based on results of joint optimization of CPU frequency and transmission power in return. It is observed from Figs. 6b and 6c that although CSA-JO could achieve very nearly the



(a) cost-number of users (b) T-number of users (c) E-number of users

Fig. 6. Comparison under increasing number of users in terms of cost, response latency and energy consumption.

same and even lower latency, its total cost still underperforms ours, since it spends much more energy on transmission and execution. Another observation can be drawn from Sections 5.2.1 and 5.2.2 that compared with other schemes, our scheme can adapt for the scenario whose scalability increases best, whether user number or task number.

5.3 Sensitivity Analysis

To explore the influence of sensitivity in our scheme, we vary control parameter ρ and weight parameter ω_i^T and ω_i^E to observe performance variation.

5.3.1 Impact of Control Parameter

In the above three subsections, we evaluate our scheme in the view of system scalability, compared with other four baseline schemes. In this subsection, we examine the performance under different values of inner control parameter ρ. Recall that ρ is a parameter which affects the performance of convergence of Markov Approximation algorithm. As reflected in Fig. 7, when we vary ρ from 1 to 100, final results achieved by our algorithm decrease. Three curves drop steeply at first and then converges smoothly. Reason behind it is that larger ρ means larger probability for system to transfer to the better state. At the initial stage, x_ϕ is relatively larger, which results in larger transition rate. It is easier for system to update to a better state. However, as state becomes better, the cost of current state decreases. Thus, it is hard for system to select a better state.

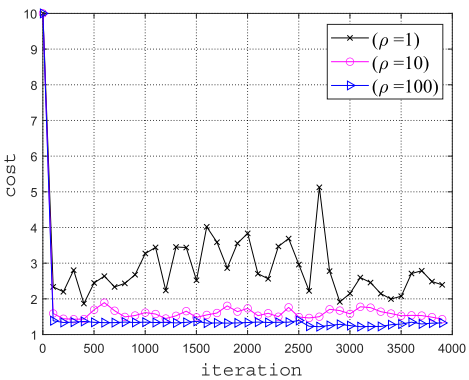


Fig. 7. Impact of control parameter.

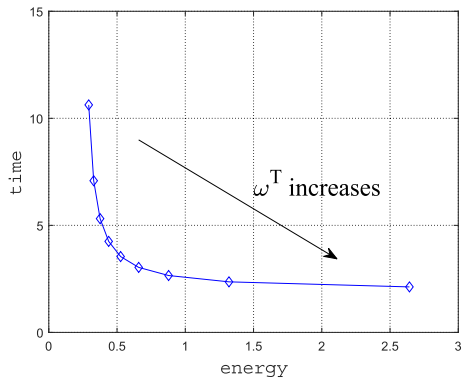


Fig. 8. Impact of weight parameter.

5.3.2 Impact of Weight Parameter

In this subsection, we examine the influence on trade-off between energy consumption and response latency brought by the weight parameter ω^T. As shown in Fig. 8, ω^T varies from 0.2 to 1.0 uniformly. We can observe from Fig. 8 that as ω^T increases, response latency naturally decreases at the same time that energy consumption becomes higher. It's not difficult to verify because latency gets more sensitive to objective $P1$ with larger ω^T. On the contrary, as ω^T decreases, that is to say ω^E increases relatively, the energy consumption is reduced.

Another observation can be obtain from Fig. 8 is that the gradient of latency with respect to ω^T is monotonically decreasing. Specifically, for the larger ω^T, the same increase results in reducing less latency and rasieing more energy consumption. It's can be seen that the most densely distributed area for ω^T is $\omega^T = 0.5$ which means the better tradeoff.

6 Discussion and Conclusion

In this paper, we investigate the joint optimization of resource allocation (CPU frequency and transmission power) and offloading strategy for composite service of mobile users, with service dependency considered. First, we introduce the computation analysis of local execution and edge computing, in terms of latency and energy consumption. Based on this, we formulate it as a MINLP whose computation complexity is non-polynomial. To deal with challengings of strong coupling between resource allocation and offloading, we assume offloading policy is given and accordingly transform it into ILP. By introducing Lagrangian multiplier method, we derive the optimal closed-form expressions of CPU frequency and

transmission power. To obtain optimal offloading policy, we adopt Markov Approximation method to update offloading decision iteratively as outer layer while the inner layer is Lagrangian multiplier method. Extensive simulations have been conducted to evaluate the performance and convergence of our scheme. In a word, our aim is providing valuable insights into inter-user dependency. This paper is a start which only accounts for master-slave model. The more complex model would be studied in the future.

Next, we extend our algorithm to a multi-round scenario where each round consists of publishing contents by master and submitting results by workers. Assume that there exist Y rounds, including Y *send* and Y *receive* processes for master, MU 1. we denote $\mathcal{Q}_{i,y}, 1 \leq y \leq Y$ to indicate the wait time of yth dependency for ith MU. When $i = 1$, we have

$$\mathcal{Q}_{1,y} = \max\{FT_{1,r_i^y-1} + TR^l_{1,r_i^y} +, TR^c_{1,r_i^y} \mathcal{T}^{max}_{1,r_i^y}\}. \tag{48}$$

Otherwise, we have

$$\mathcal{Q}_{i,y} = \max\{FT_{i,k_i^y-1} + TR^l_{i,k_i^y} + TR^c_{i,k_i^y}, \mathcal{T}^{max}_{i,k_i^y}\}, \tag{49}$$

where k_i^y and r_i^y indicate the yth *send* and *receive* tasks, respectively. If there exist workers fail to submit results, master will regard them as refusing to cooperate in that round. MU 1 could complete *receive* without their results. We define $fpred(1, r_1^y)$ to indicate the failed workers in yth round. In this case, $\mathcal{T}^{max}_{1,r_1^y}$ is equal with $\max_{i' \neq 1, m_{i',j'} \in pred(m_{1,j})} \backslash fpred(1, r_1^y) \left\{ FT_{i',j'} + \tau^{ex}_{i',j'}(1, r_1^y) \right\}$. When a is given, it is also a convex problem. The Lagrangian function can be described as

$$\begin{aligned} &L(\tau^l_{i,j}, \tau^u_{i,j}, \lambda, \mu) \\ &= \sum_{y \in Y} \mu_y \left(T_{1,r_1^y-1} + TR^l_{1,r_1^y} + TR^c_{1,r_1^y} + \mathcal{T}^{sum}_{1,r_1^y} - N\mathcal{Q}_{1,y} \right) \\ &+ \sum_{y \in Y} \lambda_y \sum_{i=2}^{N} \left(FT_{i,k_i^y-1} + TR^l_{i,k_i^y} + TR^c_{i,k_i^y} + \mathcal{T}^{sum}_{i,k_i^y} - 2\mathcal{Q}_{i,y} \right) \\ &+ \sum_{i \in N} \omega_i^T T_i + \omega_i^E E_i. \end{aligned} \tag{50}$$

Next, CSRAO can be applied to draw offloading decision. Hence, our proposed framework could solve the general inter-user problem. It's our future work to find a more low-complexity algorithm to reduce response latency and energy consumption.

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] M. Chen, B. Liang, and M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.

[3] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, Fourth Quarter 2017.

[5] S. Deng, L. Huang, J. Taheri, J. Yin, M. Zhou, and A. Y. Zomaya, "Mobility-aware service composition in mobile communities," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 47, no. 3, pp. 555–568, Mar. 2017.

[6] S. Deng, H. Wu, W. Tan, Z. Xiang, and Z. Wu, "Mobile service selection for composition: An energy consumption perspective," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 3, pp. 1478–1490, Jul. 2017.

[7] R. Kemp, N. Palmer, T. Kielmann, and H. E. Bal, "Cuckoo: A computation offloading framework for smartphones," in *Proc. 2nd Int. ICST Conf. Mobile Comput. Appl. Serv.*, 2010, vol. 76, pp. 59–79.

[8] Y. Abe, R. Geambasu, K. R. Joshi, H. A. Lagar-Cavilla, and M. Satyanarayanan, "vTube: Efficient streaming of virtual appliances over last-mile networks," in *Proc. 4th ACM Symp. Cloud Comput.*, 2013, pp. 16:1–16:16.

[9] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 207–215.

[10] Z. Xu, L. Zhou, S. C. Chau, W. Liang, Q. Xia, and P. Zhou, "Collaborate or separate? Distributed service caching in mobile edge clouds," in *Proc. 39th IEEE Conf. Comput. Commun.*, 2020, pp. 2066–2075.

[11] S. Guo, M. Chen, K. Liu, X. Liao, and B. Xiao, "Robust computation offloading and resource scheduling in cloudlet-based mobile cloud computing," *IEEE Trans. Mobile Comput.*, vol. 20, no. 5, pp. 2025–2040, May 2021.

[12] F. Wang, J. Xu, and S. Cui, "Optimal energy allocation and task offloading policy for wireless powered mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2443–2459, Apr. 2020.

[13] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 82–89, Aug. 2014.

[14] L. Yao, Y. Wang, X. Wang, and G. Wu, "Cooperative caching in vehicular content centric network based on social attributes and mobility," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 391–402, Feb. 2021.

[15] J. Yan, S. Bi, Y. J. Zhang, and M. Tao, "Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 235–250, Jan. 2020.

[16] J. Yan, S. Bi, and Y. A. Zhang, "Optimal offloading and resource allocation in mobile-edge computing with inter-user task dependency," in *Proc. IEEE Global Commun. Conf.*, 2018, pp. 1–8.

[17] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, "Self-balancing federated learning with global imbalanced data in mobile systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 59–71, Jan. 2021.

[18] J. Yao and N. Ansari, "Enhancing federated learning in fog-aided IoT by CPU frequency and wireless power control," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3438–3445, Mar. 2021.

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

[20] R. Viswanathan and P. K. Varshney, "Distributed detection with multiple sensors Part I. Fundamentals," *Proc. IEEE*, vol. 85, no. 1, pp. 54–63, Jan. 1997.

[21] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 56–69, Jul. 2006.

[22] Z. Zhong and R. Buyya, "A cost-efficient container orchestration strategy in kubernetes-based cloud computing infrastructures with heterogeneous resources," *ACM Trans. Internet Techn.*, vol. 20, no. 2, pp. 15:1–15:24, 2020.

[23] D. Cheng, X. Zhou, P. Lama, M. Ji, and C. Jiang, "Energy efficiency aware task assignment with DVFS in heterogeneous hadoop clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 1, pp. 70–82, Jan. 2018.

[24] D. Han, W. Chen, and Y. Fang, "Joint channel and queue aware scheduling for latency sensitive mobile edge computing with power constraints," *IEEE Trans. Wireless Commun.*, vol. 19, no. 6, pp. 3938–3951, Jun. 2020.

[25] 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.

[26] L. Yuan *et al.*, "CoopEdge: A decentralized blockchain-based platform for cooperative edge computing," in *Proc. Web Conf.*, 2021, pp. 2245–2257.

[27] H. Zhao, S. Deng, C. Zhang, W. Du, Q. He, and J. Yin, "A mobility-aware cross-edge computation offloading framework for partitionable applications," in *Proc. IEEE Int. Conf. Web Services*, 2019, pp. 193–200.
[28] L. Qian, Y. Wu, F. Jiang, N. Yu, W. Lu, and B. Lin, "NOMA assisted multi-task multi-access mobile edge computing via deep reinforcement learning for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5688–5698, Aug. 2021.
[29] X. Qiu, W. Zhang, W. Chen, and Z. Zheng, "Distributed and collective deep reinforcement learning for computation offloading: A practical perspective," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1085–1101, May 2021.
[30] M. Hu, Z. Xie, D. Wu, Y. Zhou, X. Chen, and L. Xiao, "Heterogeneous edge offloading with incomplete information: A minority game approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 9, pp. 2139–2154, Sep. 2020.
[31] Y. Sahni, J. Cao, L. Yang, and Y. Ji, "Multi-hop multi-task partial computation offloading in collaborative edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1133–1145, May 2021.
[32] 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.
[33] Y. Mao, W. Hong, H. Wang, Q. Li, and S. Zhong, "Privacy-preserving computation offloading for parallel deep neural networks training," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1777–1788, Jul. 2021.
[34] T. Bahreini, H. Badri, and D. Grosu, "Mechanisms for resource allocation and pricing in mobile edge computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 667–682, Mar. 2022.
[35] Y. Chen, C. Lin, J. Huang, X. Xiang, and X. Shen, "Energy efficient scheduling and management for large-scale services computing systems," *IEEE Trans. Services Comput.*, vol. 10, no. 2, pp. 217–230, Mar./Apr. 2017.
[36] C. Yi, S. Huang, and J. Cai, "Joint resource allocation for device-to-device communication assisted fog computing," *IEEE Trans. Mobile Comput.*, vol. 20, no. 3, pp. 1076–1091, Mar. 2021.
[37] Y. Wu, L. P. Qian, H. Mao, X. Yang, H. Zhou, and X. Shen, "Optimal power allocation and scheduling for non-orthogonal multiple access relay-assisted networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 11, pp. 2591–2606, Nov. 2018.
[38] I. Alqerm and B. Shihada, "Sophisticated online learning scheme for green resource allocation in 5G heterogeneous cloud radio access networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 10, pp. 2423–2437, Oct. 2018.
[39] J. Meng, H. Tan, C. Xu, W. Cao, L. Liu, and B. Li, "Dedas: Online task dispatching and scheduling with bandwidth constraint in edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 2287–2295.
[40] J. Meng, H. Tan, X. Li, Z. Han, and B. Li, "Online deadline-aware task dispatching and scheduling in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 6, pp. 1270–1286, Jun. 2020.
[41] L. Yang, J. Cao, H. Cheng, and Y. Ji, "Multi-user computation partitioning for latency sensitive mobile cloud applications," *IEEE Trans. Comput.*, vol. 64, no. 8, pp. 2253–2266, Aug. 2015.
[42] X. Lin, Y. Wang, Q. Xie, and M. Pedram, "Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment," *IEEE Trans. Services Comput.*, vol. 8, no. 2, pp. 175–186, Mar./Apr. 2015.
[43] K. Son and B. Krishnamachari, "SpeedBalance: Speed-scaling-aware optimal load balancing for green cellular networks," in *Proc. IEEE Conf. Comput. Commun.*, 2012, pp. 2816–2820.
[44] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.
[45] S. Bi, L. Huang, and Y. A. Zhang, "Joint optimization of service caching placement and computation offloading in mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 7, pp. 4947–4963, Jul. 2020.
[46] C. Lemaréchal, S. Boyd, and L. Vandenberghe, "Convex optimization," *Eur. J. Oper. Res.*, Cambridge Uni. Press, vol. 170, no. 1, pp. 326–327, 2006.
[47] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6301–6327, Oct. 2013.
[48] Z. Shao, X. Jin, W. Jiang, M. Chen, and M. Chiang, "Intra-data-center traffic engineering with ensemble routing," in *Proc. IEEE Conf. Comput. Commun.*, 2013, pp. 2148–2156.
[49] S. Zhang, Z. Shao, M. Chen, and L. Jiang, "Optimal distributed P2P streaming under node degree bounds," *IEEE/ACM Trans. Netw.*, vol. 22, no. 3, pp. 717–730, Jun. 2014.
[50] H. Zhang, "An optimized video-on-demand system: Theory, design and implementation," Ph.D. dissertation, Electr. Eng. Comput. Sci., Univ. California, Berkeley, Berkeley, CA, 2012. [Online]. Available: http://www.escholarship.org/uc/item/74k0723z
[51] R. B. Lund, "Markov processes for stochastic modeling," *J. Amer. Statist. Assoc.*, vol. 93, no. 442, pp. 842–843, 1998.
[52] P. Diaconis and D. Stroock, "Geometric bounds for eigenvalues of Markov chains," *Ann. Appl. Probability*, vol. 1, pp. 36–61, 1991.
[53] R. Bubley and M. Dyer, "Path coupling: A technique for proving rapid mixing in Markov chains," in *Proc. 38th Annu. Symp. Found. Comput. Sci.*, 1997, pp. 223–231.
[54] P. Lai *et al.*, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Proc. Int. Conf. Service-Oriented Comput.*, 2018, pp. 230–245.
[55] J. Kwak, Y. Kim, J. Lee, and S. Chong, "DREAM: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2510–2523, Dec. 2015.

Haowei Chen received the BS degree from the School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China, in 2020. He is currently working toward the PhD degree in the College of Computer Science and Technology, University of Zhejiang, China. His current research interests include edge computing and service computing.

Shuiguang Deng (Senior Member, IEEE) received the BS and PhD degrees both in computer science from Zhejiang University, China, in 2002 and 2007, respectively. He is currently a full professor with the College of Computer Science and Technology, Zhejiang University, China. He previously worked with the Massachusetts Institute of Technology, Cambridge, Massachusetts, in 2014 and Stanford University, Stanford, California, in 2015 as a visiting scholar. His research interests include edge computing, service computing, cloud computing, and business process management. He serves for the journal *IEEE Trans. on Services Computing*, *Knowledge and Information Systems*, *Computing*, and *IET Cyber-Physical Systems: Theory & Applications* as an associate editor. Up to now, he has published more than 100 papers in journals and refereed conferences. In 2018, he was granted the Rising Star Award by IEEE TCSVC. He is a fellow of IET.

Hongze Zhu received the BS degree from the School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China, in 2020. He is currently working toward the master's degree in the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. His research interests include edge computing and machine learning.

Hailiang Zhao received the BS degree from the School of Computer Science and Technology, Wuhan University of Technology, Wuhan, China, in 2019. He is currently working toward the PhD degree with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. He has been a recipient of the Best Student Paper Award of IEEE ICWS 2019. His research interests include edge computing, service computing and machine learning.

Rong Jiang received the PhD degree in system analysis and integration from the School of Software, Yunnan University, China. He is currently a deputy dean of the Institute of Intelligence Applications, distinguished professor and doctoral supervisor with the Yunnan University of Finance and Economics, Kunming, China. His current research interests include cloud computing, big data, block chain, AI application and information management, digital economy and software engineering.

Schahram Dustdar (Fellow, IEEE) is currently a full professor of computer science (Informatics) with a focus on Internet Technologies heading the Decentralized Systems Group with the TU Wien, Austria. He is chairman of the Informatics Section of the Academia Europaea (since December 9, 2016). From 2004–2010 he was honorary professor of Information Systems with the Department of Computing Science, University of Groningen (RuG), The Netherlands. From December 2016 until January 2017 he was a visiting professor with the University of Sevilla, Spain and from January until June 2017 he was a visiting professor with UC Berkeley, Berkeley, California. He is a member of the IEEE Conference Activities Committee (CAC) (since 2016), of the Section Committee of Informatics of the Academia Europaea (since 2015), a member of the Academia Europaea: The Academy of Europe, Informatics Section (since 2013). He is recipient of the ACM Distinguished Scientist award (2009) and the IBM Faculty Award (2012). He is an associate editor of *IEEE Transactions on Services Computing*, *ACM Transactions on the Web*, and *ACM Transactions on Internet Technology* and on the editorial board of *IEEE Internet Computing*. He is the editor- in-chief of *Computing* (an SCI-ranked journal of Springer).

Albert Y. Zomaya (Fellow, IEEE) is currently the chair professor of high-performance computing and networking in the School of Computer Science, University of Sydney, Australia. He is also the director of the Centre for Distributed and High Performance Computing. He published more than 600 scientific papers and articles and is a author, co-author or editor of more than 20 books. He served as the editor-in-chief of the *IEEE Transactions on Computers* (2011–2014). Currently, he is the editor in chief of *ACM Computing Surveys* and serves as associate editor for several leading journals. He delivered more than 190 keynote addresses, invited seminars, and media briefings and has been actively involved, in a variety of capacities, in the organization of more than 700 national and international conferences. He received the IEEE Technical Committee on Parallel Processing Outstanding Service Award (2011), the IEEE Technical Committee on Scalable Computing Medal for Excellence in Scalable Computing (2011), and the IEEE Computer Society Technical Achievement Award (2014). He is a chartered engineer, a fellow of AAAS, IET (U.K.), and an elected member of Academia Europaea. His research interests parallel and distributed computing and complex systems.