Task Offloading for Cloud-Assisted Fog Computing With Dynamic Service Caching in Enterprise Management Systems

Xingxia Dai, *Student Member, IEEE*, Zhu Xiao, *Senior Member, IEEE*, Hongbo Jiang, *Senior Member, IEEE*, Mamoun Alazab, John C. S. Lui, *Fellow, IEEE*, Geyong Min, *Member, IEEE*, Schahram Dustdar, *Fellow, IEEE*, and Jiangchuan Liu, *Fellow, IEEE*

***Abstract*—In enterprise management systems (EMS), augmented Intelligence of Things (AIoT) devices generate delay-sensitive and energy-intensive tasks for learning analytics, articulate clarifications, and immersive experiences. To guarantee effective task processing, in this work, we present a cloud-assisted fog computing framework with task offloading and service caching. In the framework, tasks make offloading decisions to determine local processing, fog processing, and cloud processing with the goal of minimal task delay and energy consumption, conditioned on dynamic service caching. To this end, we first propose a distributed task offloading algorithm based on noncooperative game theory. Then, we adopt the 0–1 knapsack method to realize dynamic service caching. At last, we adjust the offloading decisions for the tasks offloaded to the fog server but without caching service support. In addition, we conduct extensive experiments and the results validate the effectiveness of our proposed algorithms.**

***Index Terms*—Augmented Intelligence of Things (AIoT), enterprise management systems (EMS), game theory, service caching, task offloading.**

Manuscript received 30 March 2022; revised 20 May 2022 and 11 June 2022; accepted 22 June 2022. Date of publication 27 June 2022; date of current version 8 November 2022. This work was supported in part by the National Natural Science Foundation of China under Grant U20A20181, in part by the Key Research and Development Project of Hunan Province of China under Grant 2022GK2020, in part by the Hunan Natural Science Foundation of China under Grant 2022JJ2059, in part by the Funding Projects of Zhejiang Lab under Grant 2021LC0AB05, and in part by the Open Research Funds from Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ) under Grants GML-KF-22-22 and GML-KF-22-23. Paper no. TII-22-1346. *(Corresponding authors: Zhu Xiao; Hongbo Jiang.)*

Xingxia Dai is with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan 410082, China, and also with the Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen 518060, China, (e-mail: xingxdai718@gmail.com).

Zhu Xiao and Hongbo Jiang are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan 410082, China (e-mail: zhxiao@hnu.edu.cn; hongbojiang2004@gmail.com).

Mamoun Alazab is with the College of Engineering, IT and Environment, Charles Darwin University, Darwin, NT 0810, Australia (e-mail: mamoun.alazab@cdu.edu.au).

John C. S. Lui is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong (e-mail: cslui@cse.cuhk.edu.hk).

Geyong Min is with the Department of Mathematics and Computer Science, University of Exeter, EX4 4PY Exeter, U.K. (e-mail: gmin@exeter.ac.uk).

Schahram Dustdar is with TU Wien, 1040 Vienna, Austria (e-mail: dustdar@infosys.tuwien.ac.at).

Jiangchuan Liu is with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada (e-mail: jcliu@cs.sfu.ca).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2022.3186641.

Digital Object Identifier 10.1109/TII.2022.3186641

I. Introduction

ENTERPRISE management systems (EMS) as the combination of industrial Internet of Things and Intelligence of Things, has drawn ever-increasing attention from academia to industry, boosting the development of smart society [1], [2]. To improve the interrelation with human beings, augmented intelligence is used to develop a human-centered partnership model for smart EMS. The augmented Intelligence of Things (AIoT) supports learning analytics, articulate clarifications and immersive experiences. To this end, AIoT devices generally generate massive data and require fast task processing. For example, to implement real-time vision applications in EMS, AIoT devices need to capture massive video data, then perform complicated task processing for each frame under strict delay constraints [3], [4]. However, due to limited local computing capability and energy budget, operating these applications in AIoT devices will inevitably incur large task computing delay and thus suffer from degraded processing efficiency.

Fog computing emerges as a promising solution, by which sufficient computation resources are pushed from a cloud server to the network edge [5]–[8]. This enables fog servers the provision of computing resources in the manner of proximity to AIoT devices. In this way, delay-sensitive and energy-intensive tasks can be offloaded from AIoT devices to fog servers, thereby facilitating low task delay and energy consumption [9].

A problem arising in fog computing is the service caching issue [10], [11]. Specifically, task processing requires computation services (e.g., video streaming analysis) and related databases/libraries (e.g., online deep learning frameworks). Without service support, the tasks cannot be processed on the platform. But, a fog server is unable to cache overall services requested by the diverse tasks due to limited storage space. As a result, if the requested services are not available in the fog

server, these tasks are not supported by fog processing. Thus, how to make use of the limited fog storage space is a question worth considering for efficient task offloading.

However, previous related studies fail to consider the service caching issue effectively in task offloading [12]–[16]. In these works, a fog server is assumed to have infinite computation services, or assumed to follow fixed service caching strategies in task offloading. Clearly, the former assumption does not hold in the real world, since fog server is inherently constrained by limited storage space. Under the unsuitable assumption, task offloading is easily trapped by suboptimal offloading solutions. Additionally, the latter assumption inevitably degrades task offloading efficiency. Specifically, AIoT devices generate diverse tasks, and the requested computation services are changing. If the fog server follows a fixed service caching strategy, its available computation services will keep invariable. As a result, fog offloading is only available for specific tasks during the whole offloading. The fixed service caching degrades offloading efficiency, especially when the tasks are requested by AIoT devices frequently while the computation services are not cached in the fog server.

As a remedy, several studies jointly consider task offloading and service caching issues, such as [17]–[19]. However, they do not consider the issues for cloud-assisted fog computing. As such, tasks not supported by fog computing have to be processed locally, and thus incur large computation delay, especially for computing-intensive tasks. Actually, cloud computing, due to abundant resources, enables low computation delay, especially for tasks with large computation workloads and small data bits. Thus, cloud-assisted fog computation has been proposed in [6], [20]–[24]. But, these works neglect service caching issues for cloud-assisted fog computing.

To bridge the gap, we jointly formulate the task offloading and service caching problems for cloud-assisted fog computing in EMS. To find out feasible solutions to the problem, we are facing several challenges: 1) Service caching couples with task offloading decisions. Tasks are unable to seek fog computing when the computation services are not cached in the fog server; and in return, task offloading results reflect the performance of service caching. To achieve effective task processing, the interplay between offloading decisions and service caching needs to be taken into consideration. 2) The formulated problem is non deterministic polynomial (NP)-hard since both the offloading decisions and service caching strategies are discretized and combinational. Additionally, AIoT devices are distributed and deployed in the real world, and the global information is hard to obtain for decision-making due to lacking a centralized controller.

To address the aforementioned challenges, we derive an approach which jointly considers task offloading and service caching for cloud-assisted fog computing. Specifically, tasks can be offloaded to a fog server for less task delay and local energy consumption. Then, the fog server seeks dynamic service caching for a cloud server based on the task popularity. Due to the constrained fog storage space, the fog server caches limited computation services. When the requested service caching is not cached in the fog server, the tasks can be offloaded to the cloud server or processed locally. Guided by the approach, both offloading efficiency and service caching are emphasized for cloud-assisted fog computing. Our contributions are summarized as follows.

1) We jointly consider task offloading and service caching issues for cloud-assisted fog computing in EMS. To achieve minimal system cost, i.e., the weighted sum of task delay and local energy consumption, tasks make offloading decisions. Considering limited fog storage, a dynamic service caching is derived, where the fog server flexibly requests computation services from the cloud server based on the task popularity.
2) We propose an algorithm supporting task offloading and service caching for cloud-assisted fog computation (TO&SC-CF), detailed in Section IV. Specifically, we first propose a distributed task offloading algorithm based on noncooperative game theory. Based on the offloading decisions, we adopt the 0–1 knapsack method to realize dynamic service caching. Guided by the service caching, we adjust the offloading decisions for tasks offloaded to the fog server and without service supports.
3) We conduct extensive experiments, detailed in Section V, to validate the effectiveness of our proposed algorithms. The simulation results demonstrate the superiority of the proposed algorithm compared with other algorithms under various system parameters, such as required CPU cycles, task data bits, task number, and fog storage units.

The rest of the article is organized as follows. Section II presents the related works followed by the system model and problem formulation in Section III. Section IV is TO&SC-CF. In Section V, we conduct evaluations. Finally, Section VI concludes this article.

II. Related Works

Fog computing has emerged as an attractive paradigm for task processing, by which computing-intensive tasks can be offloaded to the fog server for pursuing less task delay and local energy consumption [12]–[16]. In [12], the authors formulated two optimization problems in fog computing, where smart terminals enable task offloading to the nearby fog server. By jointly considering communication and computation states, the optimal offloading solutions can be found with the goal of minimal local energy consumption. Based on the solutions, the authors study a task assignment problem to further minimize energy consumption. In [13], the authors considered offloading the computation tasks to an edge server or a third-part fog node. Based on the offloading requests, both computation and communication resources are managed in the fog computing system, aiming at the maximum network management profit. In [14], the authors performed a dynamic resource allocation framework in the fog computation system, where users enable to offload tasks to the fog server and need to pay for the allocated computation resources. The problem is formulated as a mixed-integer nonlinear programming problem, targeting the minimal task delay, energy consumption, and price cost. In [15], the authors studied the provisioning problem of the virtualized

network function services in fog computing. By considering the mobility and delay requirements, an optimization problem maximizing network utility and throughput is formulated. In [16], the authors proposed a secure task-offloading framework, where tasks are offloaded to the fog node. The selected node enables secure task offloading that it offloads tasks to a neighboring fog node based on smart contracts.

To maintain effective processing, service caching deserves particular attention in task offloading. There are several studies that jointly consider task offloading and service caching [17]–[19]. In [17], the authors formulated a mixed-integer nonlinear programming problem, in which a mobile user offloads its tasks to the fog server and the server dynamically caches services for task processing. By optimizing offloading decisions, service caching and resource allocation, minimal task delay and local energy consumption can be realized. In [18], the authors studied the dependent task offloading problem under the limited fog storage constraint. Only services are cached in the fog server, the tasks enable fog processing. To solve the problem, the authors design an efficient convex programming algorithm to optimize task offloading and service caching. In [19], the authors addressed task offloading and service caching problems under the constraints of communication resources and task delay, targeting maximum system utility. Based on this, the authors derive effective algorithms under the circumstances of homogeneous and heterogeneous service caching.

Compared with a cloud server, the fog server only has limited resources. Consider this, several works investigate cloud-assisted fog computation [6], [20]–[24]. In [6], the authors designed a cloud-assisted fog computing framework, where tasks are mostly offloaded to the fog node. Considering limited fog resources in computation and storage, the selected fog node can be further offloaded tasks to its neighboring fog nodes and/or the cloud server. In this framework, the authors investigate a delay-driven task offloading problem. In [20], the authors considered a task offloading and resource allocation problem in a fog-cloud computing environment. Based on the Lyapunov optimization technique, the authors propose an online algorithm to solve the problem, aiming at minimizing the average task delay. In [21], the authors designed a framework with multiple devices, multiple fogs servers, and a cloud server, targeting minimal task delay and local energy consumption. On this basis, the authors propose a placement technique algorithm and a lightweight prescheduling algorithm for concurrent applications. In [22], the authors proposed a mincost offloading partitioning algorithm. The algorithm enables the minimal partitioning cost under different cost models and mobile environments. This is achieved by optimizing application partitioning decisions to determine whether to process the applications locally or offload them to cloud and fog servers. In [23], the authors proposed an effective task offloading solution for real-time traffic management in fog-based Internet of vehicles (IoV) systems. To this end, the authors leverage queue theory to model vehicle states and derive an approximate method to achieve task offloading optimization. In [24], the authors considered an online deadline-aware task dispatching and scheduling in fog computing. By optimizing the network bandwidth and computing resources, the authors aim to the maximum number of tasks completed within the deadline.

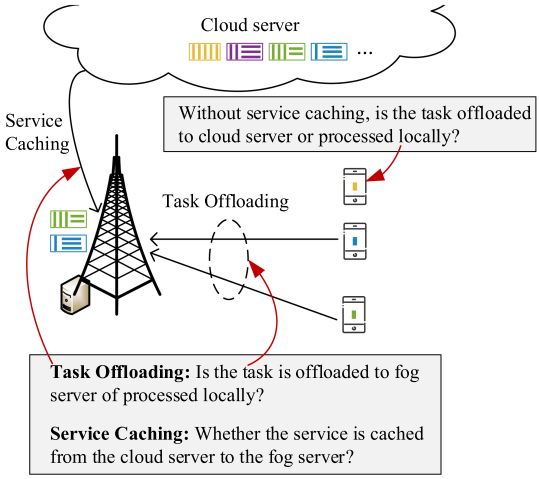


Fig. 1. System model. Combining TO&SC, tasks make offloading decisions.

As these works mentioned, fog offloading and service caching are both essential for effective task processing. However, most of the existing works consider task offloading and service caching separately, which is not realistic and easily trapped by supoptimal offloading and caching solutions. Additionally, existing works mostly neglect tasks for which the requested services are not available in fog computing or assume these tasks are processed locally without cloud-assisted offloading. This leads to degraded offloading efficiency. Different from these works, we study task offloading and service caching jointly in this work. Furthermore, we present a cloud-assisted framework that efficiently overcomes deficient fog service resources. Guided by this, we derive a joint task offloading and service caching algorithm for cloud-assisted fog computation.

III. System Models and Problem Formulation

Fig. 1 presents the system model of (TO&SC-CF. The model comprises a cloud server, a fog server and multiple AIoT devices. Each AIoT device generates one task to be processed. Let $\mathcal{N} = \{1, \ldots, N\}$ denote the task set. In EMS, AIoT devices typically offload their tasks to the fog server since fog computing facilitates low computation delay while keeping small transmission delay. As such, in *task offloading* phase, AIoT devices either process their tasks locally or offload tasks to the fog server. In addition, the fog server needs to equip diverse computation services to support task processing. However, constrained by the storage space, the fog server only can cache limited computation services. Different from fog computing, cloud computing has abundant computation services. Thus, we derive *dynamic service caching* for fog computing, where the fog server enables to flexibly request computation services from the cloud server based on the task popularity. While dynamic caching ensures that popular computation services are cached, several services are

TABLE I
SUMMARY OF THE KEY NOTATIONS

Notations	Definitions
$\mathcal{N}$	The task set
$\mathcal{K}$	The service set
$\mathbf{x}_n$	The offloading decision of task n
T_n^l	The local processing delay of task n
c_n	The required CPU cycles of task n
f_n	The local CPU capabilities
E_n^l	The local energy consumption of task n
$T_{trans}^{f,n}$	The transmission delay for fog processing
b_n	The data bits of task n
r_n^f	The transmission rate between task n and the fog server
$E_{trans}^{f,n}$	The transmission energy consumption for task n
p_n^f	The transmission power of task n
f_n^f	The allocated fog computing resources for task n
r_n^c	The transmission rate between task n and the cloud server
α_n^f	The service caching decision of the fog server

still not available in the fog server. Once the requested services are not cached in the fog server, AIoT devices need to adjust the offloading decisions, i.e., to determine whether the tasks are processed locally or offloaded to the cloud server. We list the key notations as Table I for better readability.

A. Task Offloading Model

The offloading decision of task n is denoted as a vector $\mathbf{x}_n = \{x_n^l, x_n^f, x_n^c\}$, in which each element is a binary variable.

1) Local Processing: When $x_n^l = 1$ and $x_n^f = x_n^c = 0$, task n is processed locally. The local computation delay is expressed as

$$T_n^l = c_n / f_n \tag{1}$$

where c_n denotes the required CPU cycles of task n and f_n is the local CPU cycle frequency. Local processing also incurs energy consumption

$$E_n^l = \varpi c_n (f_n)^2 \tag{2}$$

where ϖ is the energy coefficient related to the chip architecture of the AIoT device.

2) Fog Processing: When $x_n^f = 1$ and $x_n^l = x_n^c = 0$, task n will be offloaded to the fog server for processing [25]. The task offloading goes through three phases: 1) data transmission; 2) task processing; 3) result feedback.

When task n is transmitted from the AIoT device to the fog server, the transmission delay is

$$T_{\text{trans}}^{f,n} = b_n / r_n^f \tag{3}$$

where b_n represents the data bits of task n, r_n^f reflects the transmission rate between the fog server and the AIoT device generating task n [26]. Given the transmission power p_n^f of the AIoT [27], [28], we obtain the transmission energy consumption

$$E_{\text{trans}}^{f,n} = p_n^f T_{\text{trans}}^{f,n}. \tag{4}$$

After receiving the offloaded tasks from AIoT devices, the tasks will be processed by the fog server. We use f_n^f to denote the allocated fog computing resources for processing task n. Guided by this, we calculate the computing delay of task n for fog processing

$$T_{\text{comp}}^{f,n} = c_n / f_n^f. \tag{5}$$

In this manner, the computation energy consumption is conducted by the fog server [29]. Note that, in this work, we focus on the local energy consumption rather than that of fog. Additionally, to complete the fog offloading, the processing results are required to feedback from the fog server to the AIoT devices. Since the data bits of the results are much smaller than that of offloaded tasks, we ignore the delay and energy consumption of result feedback in task offloading [30].

3) Cloud Processing: When $x_n^c = 1$ and $x_n^l = x_n^f = 0$, task n will be offloaded to the cloud server for processing. Different from fog offloading, cloud server has more powerful computing capabilities [18], and thus we overlook the cloud processing delay and focus on data transmission. The transmission delay for pursuing cloud processing is

$$T_{\text{trans}}^{c,n} = b_n / r_n^c \tag{6}$$

where r_n^f is the transmission rate between the could server and AIoT generating task n. Correspondingly, we obtain the transmission energy consumption, expressed as

$$E_{\text{trans}}^{c,n} = p_n^c T_{\text{trans}}^{c,n}. \tag{7}$$

B. Service Caching Model

In EMS, the dedicated set of computation services and their related databases/libraries are cached in the fog server to support task processing. Due to the constrained storage space, a fog server caches limited computation services. But, AIoT devices generate different tasks, and thus the requested computation services are changing. If the fog server follows a fixed caching strategy, its available computation services will be invariable. As a result, when the fog server does not cache the computation services initially, fog offloading is not supported for the specific tasks during the whole offloading. The fixed service caching inevitably degrades offloading efficiency, especially when the tasks are requested by AIoT devices frequently while the computation services are not cached. To bridge the gap, inspired by [31], we design a *dynamic* service caching scheme in this work. Specifically, the fog server flexibly requests computation services from the cloud server based on diverse tasks. In this way, service caching policies can be dynamically adjusted rather than keep fixed, which facilitates task offloading efficiency.

As presented in Fig. 2, we elaborate on the operational flow of the dynamic service caching as follows.

1) For tasks to be offloaded to the fog server, the AIoT devices first inform the fog server of the requested service information before task transmission.
2) After receiving the information, the fog determines the popularity of each service based on the requested number. Services with large numbers have high popularity, and thus hold more chances to be cached in the fog server.
3) Considering the popularity and limited storage space, the fog server downloads services from the cloud to cache. We introduce a binary variable $\alpha_n^f = \{0, 1\}$ to illustrate whether the services requested by task n is cached ($\alpha_n^f =$

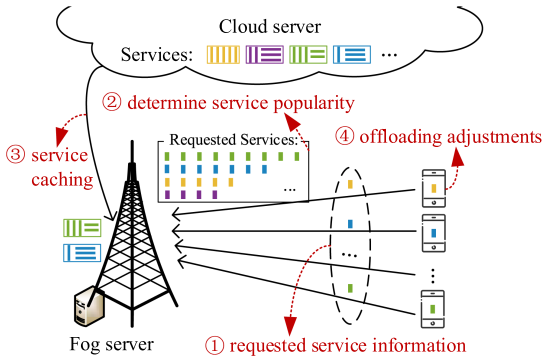


Fig. 2. Dynamic service caching in task offloading.

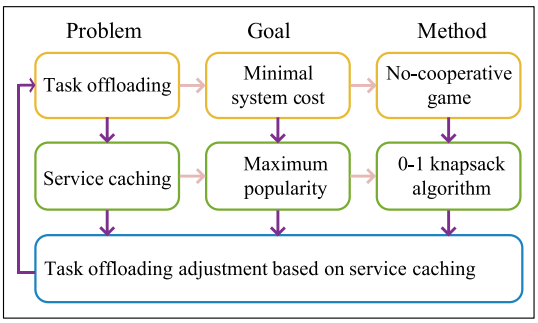


Fig. 3. Workflow of the TO&SC-CF algorithm.

1) or not ($\alpha_n^f = 0$) in the fog server. If the service is cached in the fog server, the tasks can be processed in the fog.
4) When the requested services are not cached in the fog server, the AIoT devices have to adjust the offloading decisions for these tasks, i.e., to determine whether to offload the task to the cloud or process it locally. It is noted that the service download from cloud to fog and the task transmission from AIoT devices to the fog server are concurrent. We assume that the download delay is no more than the transmission delay due to the high-speed wired links between the fog and cloud. As such, this dynamic service caching will not incur additional delay.

C. Problem Formulation

Tasks to be offloaded are delay-sensitive and energy-intensive, which motivate us to achieve the minimal system cost, i.e., the weighted sum of task delay and local energy consumption. Considering the allocated CPU cycles, data transmission rate, and power, each AIoT device makes offloading decisions to determine whether to process its task locally or at a fog server. Due to limited storage capabilities, the fog cannot cache overall computation services requested by tasks. For tasks of which the services are not cached in the fog server, these tasks need to adjust the offloading decisions, i.e., to determine whether to process its task locally or at the cloud server for less system cost.

Combining offloading decision and service caching, we obtain the delay of task n, expressed as

$$T_n = x_n^l T_n^l + \alpha_n^f x_n^f (T_{\text{trans}}^{f,n} + T_{\text{comp}}^{f,n}) + x_n^c T_{\text{trans}}^{c,n}. \tag{8}$$

The local energy consumption of task n, expressed by

$$E_n = x_n^l E_n^l + \alpha_n^f x_n^f E_{\text{trans}}^{f,n} + x_n^c E_{\text{trans}}^{c,n}. \tag{9}$$

Mathematically, we formulate the problem as

$$\min_{\mathbf{x}_n} \sum_{n\in\mathcal{N}} \beta T_n + (1-\beta) E_n \tag{10}$$

$$\text{s.t. } \alpha_n^f \in \{0,1\} \tag{11a}$$

$$s_n^f \le s^{\max} \tag{11b}$$

$$x_n^c, x_n^l, x_n^f \in \{0,1\} \tag{11c}$$

$$f_n \le f_n^{\max}, f_n^f \le f^{\max} \tag{11d}$$

where the constraint (11a) implies that a service only has two states: 1) cached; 2) not in the fog server, the constraint (11b) reflects that the cached services cannot exceed the storage space of the fog server, the constraint (11c) denotes that each task can be only offloaded to a single processor for continuity, and the constraint (11d) indicates that the allocated computing resources need to be less than the maximum computing resources of the AIoT device and fog server.

However, it is not a trivial issue to find the optimal solution to the formulated problem. Firstly, this is a nonconvex problem, since the offloading decisions and service caching strategies are all discretized solutions. Worse still, there are combinational features of task offloading and service caching, making the formulated problem difficult to solve directly. Secondly, the AIoT devices are hard to obtain global information in the real world, since the centralized controller is generally lacking in the real world.

IV. Task Offloading and Service Caching for Cloud-Assisted Fog Computing

To tackle the difficulties in solving the formulated problem, in this section, we derive a joint TO&SC-CF. As presented in Fig. 3, we formulate a noncooperative game to determine task offloading decisions firstly. Then, we adopt the 0–1 knapsack method to determine the dynamic service caching strategies. Guided by the caching solutions, we adjust the task offloading decisions.

A. Noncooperative Game for Task Offloading

In EMS, AIoT devices typically offload their tasks to the fog server since fog computing facilitates low computation delay while keeping small transmission delay. As such, in the task offloading phase, AIoT devices either process their tasks locally or offload tasks to the fog server. Considering scattered locations of AIoT devices, we formulate a noncooperative game for achieving effective task offloading. In the game, each AIoT makes offloading decisions without requiring a centralized controller. This distributed method guarantees the system utility while paying attention to the individual interests [32], [33]. We present the game as

$$\mathbb{G} = (\mathcal{N}, \mathcal{A}, \mathcal{U}(\mathcal{A})) \tag{12}$$

Algorithm 1: Task Offloading Algorithm.

Input : $b_n, c_n, f_n, f_n^f, r_n^f, r_n^c, p_n^f$ and p_n^c.
Output: The optimal task offloading strategy $\mathcal{A}_*$.

```
Compute the utility value for each feasible strategy based on Equation (14).
Feedback to the last step until we obtain the utilities of all strategies.
Choose a current strategy A_i randomly.
while A_i ∈ A do
    Calculate the regret degree for strategy A_i based on Equation (15).
    if P(A_i) > 0 then
        The strategy A_i becomes one of the candidate strategies.
    end
    Update the current strategy by choosing a new strategy from the candidate strategies.
    Retain the optimal strategy A_* with the maximun U(A_*) up to now.
end
```

where $\mathcal{N}$ represents the finite task set, $\mathcal{A}$ denotes all feasible task offloading decision set, and $\mathcal{U}(\mathcal{A})$ indicates the utilities performed by the task offloading decisions $\mathcal{A}$. Give the ith allocation strategy of $\mathcal{A}_i$, we define the utility function as

$$U(\mathcal{A}_i) = F - f(\mathcal{A}_i), i \in \mathcal{A} \tag{13}$$

where F is a large value to ensure positive $U(\mathcal{A}_i)$, $f(\mathcal{A}_i)$ is the optimization objective in (10), expressed as

$$f(\mathcal{A}_i) = \beta T(\mathcal{A}_i) + (1-\beta)E(\mathcal{A}_i). \tag{14}$$

$T(\mathcal{A}_i)$ and $E(\mathcal{A}_i)$ are the total task delay and local energy consumption performed by the policy $\mathcal{A}_i$. Note that the expressions of (10) and (14) reflect the weighted sum of task delay and local energy consumption from the task and the offloading policy perspectives, respectively.

Based on the "regret-matching" approach in the noncooperative game, we attempt to find out the optimal offloading strategy $\mathcal{A}_*$ (i.e., achieving minimal system cost) by maximizing the utility value $U(\mathcal{A}_*)$. We use $R(\mathcal{A}_i, \mathcal{A}_j)$ to measure the regret degree for applying the strategy $\mathcal{A}_j$ instead of $\mathcal{A}_i$, expressed as

$$R(\mathcal{A}_i, \mathcal{A}_j) = \max\{U(\mathcal{A}_i, \mathcal{A}_{-i}) - U(\mathcal{A}_j, \mathcal{A}_{-j}), 0\}. \tag{15}$$

If $R(\mathcal{A}_i, \mathcal{A}_j) > 0$, the strategy $\mathcal{A}_j$ will be replaced by $\mathcal{A}_i$ to improve the utility value. As such, the probability distributions of strategy $\mathcal{A}_i$ and $\mathcal{A}_j$ at the next iteration are

$$P(\mathcal{A}_i) = \frac{1}{\tau} R(\mathcal{A}_i, \mathcal{A}_j), i \neq j \tag{16}$$

$$P(\mathcal{A}_j) = 1 - \sum_{\mathcal{A}_i \in \mathcal{A}} P(\mathcal{A}_i), j \neq i \tag{17}$$

where τ is a large value for ensuring positive $P(\mathcal{A}_j)$.

The proposed task offloading algorithm is summarized in Algorithm 1. After I times iteration, we obtain the time complexity of Algorithm 1 is $\mathcal{O}(AI)$, where $A =\mid \mathcal{A} \mid$ is the total number of the feasible offloading strategies.

Let $\rho_\xi(\cdot)$ denote the empirical distribution for all feasible allocation policies of $\mathcal{A}$ for the fog computation resources. $S_\xi(\mathcal{A}_i)$ is the occurrence for the strategy $\mathcal{A}_i$ up to ξ iterations. As such, we obtain the adopted frequency of the strategy $\mathcal{A}_i$ during the period

$$\rho_\xi(\mathcal{A}_i) = \frac{S_\xi(\mathcal{A}_i)}{\xi}. \tag{18}$$

Based on the analysis in [32] and [34], we conclude that $\rho_\xi(A_i)$ converges to correlated equilibrium when ξ intends to infinity.

B. 0–1 Knapsack-Based Service Caching

Service caching is critical for avoiding infeasible task offloading decisions [18], [35]. In EMS, the fog server has constrained storage capabilities, and thus enables several dedicated and limited service supports. If the services frequently requested by AIoT devices are not available in the fog server, the efficiency of fog processing will be significantly degraded.

To guarantee effective task offloading, we propose a dynamic service caching algorithm to determine which services need to be cached in the fog server. The algorithm enables the fog server to flexibly requests computation services $k \in \mathcal{K}$ from the cloud server based on diverse tasks. Let $\lambda_n^k \in \{0, 1\}$ denote the requested service by task n. When $\lambda_n^k = 1$, service k is requested by task n; if $\lambda_n^k = 0$, service k is not available in the fog server. On this basis, we define L_k as the popularity degree of computation service k based on the requested times

$$L_k = \sum_{n \in \mathcal{N}} \lambda_n^k. \tag{19}$$

In addition, each service occupies s_k storage space for the cache. The total cached services in the fog server cannot exceed the maximum fog storage space $s_{\text{fog}}^{\max}$. As such, dynamic service caching aims to achieve maximum service popularity under the fog storage constraint. These features enable us to formulate the service caching as a 0–1 knapsack problem. Specifically, if a service occupies excessive storage space, the service will not be cached in the fog server. When a service satisfies the fog storage constraint and produces larger popularity, it will be cached in the fog server. On this basis, we derive the 0–1 knapsack-based dynamic service caching algorithm detailed in Algorithm 2. The time complexity is $\mathcal{O}((S+1)(K+1))$, where $S =\mid s_{\text{fog}}^{\max} \mid$ is the maximum fog storage capabilities and $K =\mid \mathcal{K} \mid$ is the total number of the requested services.

C. TO&SC-CF Algorithm

After determining service caching policies, we need to adjust the task offloading decisions. The reason is that task offloading decisions may not be supported by service caching policies. Specifically, tasks are offloaded to the fog server or processed locally with the goal of the minimal system cost after implementing Algorithm 1. Note that the offloading decisions are based on the infinite computation services in the fog server. However, the fog server has limited storage space and the limited storage hinders the cache for the overall requested computation service. As such, we derive Algorithm 2 to implement dynamic service

Algorithm 2: Dynamic Service Caching Algorithm.

Input : s_k, L_k and s_{fog}^{max}.
Output: The optimal fog service caching policies, $b_n^k, k \in \mathcal{K}, n \in \mathcal{N}$.

Record the current remaining fog storage space s_r.
for *k-th service* **do**
  if $s_k > s_r$ **then**
    k-th service will not be cached.
  end
  else
    if *Service k prodeces a larger popularity* **then**
      k-th service will not be cached.
    end
    else
      k-th service will be cached.
      Update the fog remaining storage space.
    end
  end
end

Algorithm 3: TO&SC-CF Algorithm.

Input : $c_n, d_n, f_n, f_n^f, r_n^f, r_n^c, p_n^f, p_n^c, s_k, L_k, s_{fog}^{max}$.
Output: The final task offloading strategy.

Offload tasks based on Algorithm 1.
Cache services based on Algorithm 2.
for *n-th task* **do**
  if $x_n^f = 1$ **then**
    When $b_n^k = 1$, task n will be offloaded to the fog server.
    When $b_n^k = 0$, task n will be processed locally or offloaded to the cloud server.
  end
end

caching, where a fog server flexibly caches computation services from the cloud server based on the task popularity. To guarantee effective task offloading, the computation services frequently requested by AIoT devices have a large chance to be cached in the fog server. While dynamic caching ensures that popular computation services are cached, several services are still not available in the fog server. Suppose a task should be processed by the fog server guided by Algorithm 1, but the requested services are not available in the fog server after implementing Algorithm 2. In this case, the task has to adjust its offloading decisions since fog offloading is not supported for the task. As a result, the task needs to change its offloading decisions, i.e., determine whether to process the task locally or offload the task to the cloud server targeting less system cost.

To achieve effective TO&SC-CF in EMS, we propose the TO&SC-CF algorithm, detailed in Algorithm 3. The algorithm decouples task offloading decisions and service caching policies, while ensuring task processing with minimal system cost under the service caching constraint. The time complexity of Algorithm 3 is $\mathcal{O}(AI + (S+1)(K+1) + N)$, where $N = \mid \mathcal{N} \mid$ is the total number of the tasks.

TABLE II
PARAMETER SETTINGS

Parameter description	Value
The number of the cloud server	1
The number of the fog server	1
The number of AIoT devices	5
The number of services	3
The data bits of each task	[0.1, 2] MB
The required CPU cycles of each task	[0.2, 4] GHZ
The computing capabilities of the fog server	[5, 15] GHZ
The computing capabilities of an AIoT device	[1, 2] GHZ
The data transmission rates to the fog server	[2, 3] Mb/s
The transmission power	0.1 Watt
The energy coefficient ϖ [17]	10^{-26}

V. PERFORMANCE EVALUATION

In this section, we conduct experiments to evaluate the proposed TO&SC-CF algorithm.

A. Simulation Setup

We conduct experiments on a desktop computer with an 11th Gen Intel(R) Core(TM) i7-11700F @2.50 GHz, 16 GB memory and Win10 OS. In our experiments, we consider five AIoT devices for cloud-assisted fog computing. Each AIoT device generates a task to be processed. The parameter settings are listed, shown in Table II. We compare the proposed algorithm with the following algorithms.

1) *Task offloading with dynamic service caching (TO&SC)* [31]: The algorithm enables offloading decisions based on dynamic service caching in fog computing, without considering the cloud-assisted fog computing scheme.
2) *Task offloading for cloud-assisted fog computing (TO&CF)* [36]: The algorithm follows fixed service caching rules. When the requested services are not cached in the fog server, the tasks can be offloaded to the cloud server for less task delay and local energy consumption.
3) *Task offloading in fog computing (TO) [37]:* Guided by the fixed service caching scheme, the algorithm makes task offloading decisions. If the requested services are not available in the fog server, the tasks have to be processed locally.
4) *Local task processing (LP):* Offloading is not available and tasks are processed locally in the algorithm.

B. Comparison Analysis

1) Comparison Analysis on Required CPU Cycles: Fig. 4 shows the comparison of system cost between different algorithms under various CPU cycles. As the required CPU cycles grow, the system cost increases. The reason is that large CPU requirements enlarge task computation delay and computing energy consumption. Additionally, we noticed that the growth rate of LP, TO, and TO&SC are similar with increasing CPU cycles. The reason is that the service with varying CPU cycles does not have to be cached in the fog server in our setting, then TO and TO&SC need to process the task with large CPU cycles locally.

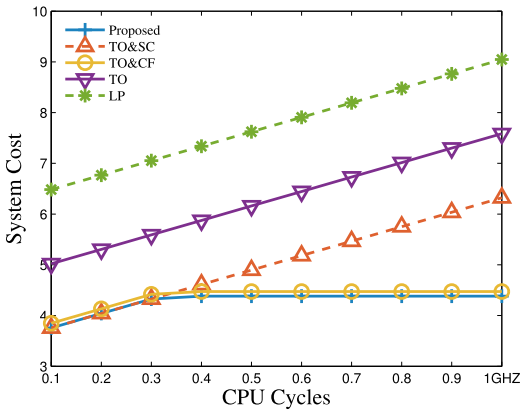


Fig. 4. Performance of different algorithms under various CPU requirements.

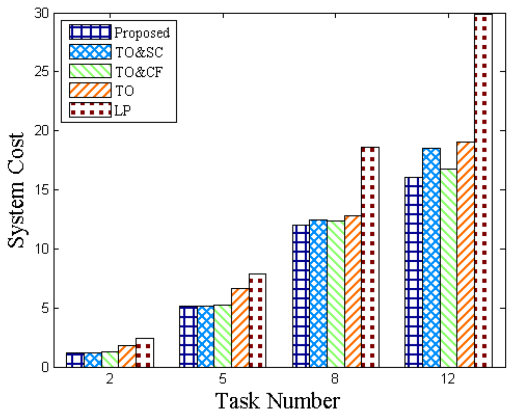


Fig. 6. Performance of different algorithms under different task number.

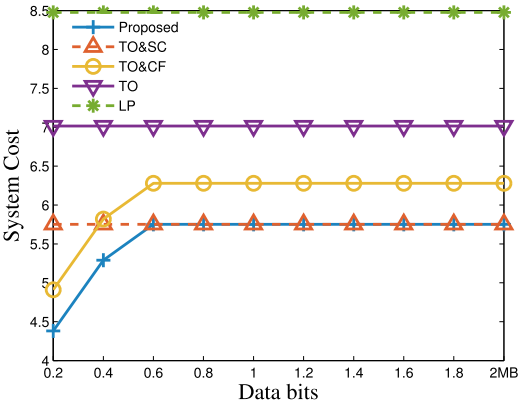


Fig. 5. Performance of different algorithms under different task data bits.

Benefitting from the cloud-assisted scheme, the proposed algorithm and TO&CF enable to offload the computing-intensive tasks to the cloud server. Since the system cost is determined by transmission energy consumption in cloud processing, the system cost keeps fixed with different CPU cycles when the task is offloaded to the cloud server. Furthermore, the system cost conducted by the proposed algorithm and TO&CF is similar for small fog storage units in the setting. Once we enlarge the storage, the performance difference between these two algorithms becomes even more pronounced, as shown in Fig. 7.

2) Comparison Analysis on Task Data Bits: Fig. 5 shows the comparison of system cost between different algorithms under various task data bits. Task data bits affect the system cost by changing transmission delay and energy consumption. Large data bites incur large system cost. But we find that the system cost of LP, TO, and TO&SC remain fixed. The reason is that LP is independent of task data bits; for TO and TO&SC, they will process the tasks locally when the service requested by the task with various data bits is not cached in the fog server. Since TO&SC enables dynamic service caching, this ensures frequently requested services have a high priority in fog caching, and thus TO&SC achieves less system cost than TO; fog processing is supported in TO, then TO performs better than LP. For the proposed algorithm and TO&CF, the task will be offloaded to the cloud server when the data bits are small and the requested service is not cached in the fog server. Larger data bits incur more transmission energy consumption, thus increasing the system cost. As the data bits grow, the system cost conducted by the cloud server will exceed that conducted by local processing. In this case, the task will be processed locally, and thus the system cost conducted by the proposed algorithm is near to that conducted by TO&SC. Furthermore, TO&CF follows fixed service caching rules, and several frequently requested services may not be cached in the fog server. Therefore, TO&CF holds a large convergent value compared with the proposed algorithm.

3) Comparison Analysis on Task Number: Fig. 6 shows the impact of task number on system cost. More tasks lead to larger task delay and energy consumption and thus increasing the system cost. Since LP does not support fog processing, the system cost sharply increases as the task number grows. TO, due to fixed service caching and without the cloud-assisted scheme, its system cost is more than that of TO&CF, TO&SC, and the proposed algorithm. The proposed algorithm follows the dynamic service caching in cloud-assisted fog computing networks, and thus produces less system cost.

4) Comparison Analysis on Fog Storage Space: Fig. 7 shows the comparison of the system cost between different algorithms under various fog storage spaces. Large storage units mean the fog server enables to delivery computation services for more tasks. The system cost of LP remains fixed with changing fog storage units due to local processing. Different from the LP, the remaining algorithms involve fog processing. As the storage increases, their system cost decreases. Furthermore, since TO and TO&SC do not consider cloud processing, fog storage units have a more significant impact on the system costs. When the fog has cached the requested services, the system costs remain fixed. In addition, we noticed that the system cost performed by the proposed algorithm and TO&CF decreases slowly. The reason is that these two algorithms can offload tasks

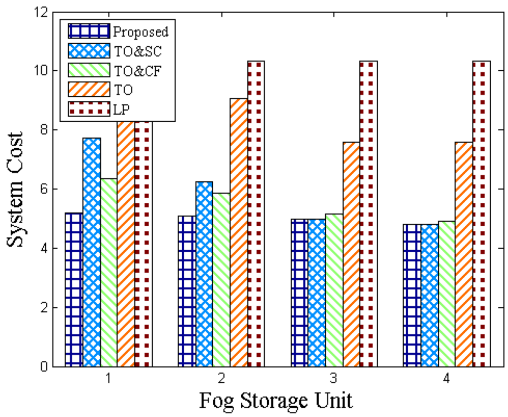


Fig. 7. Performance of different algorithms under different fog storage units.

to the cloud server when the requested services are not cached in the fog server. When the fog storage is small and the task CPU is large, TO&SC typically performs worse than TO&CF; when the fog storage is large, TO&SC typically performs better than TO&CF. Furthermore, since the fog server incapacitates dynamically adjusting service caching strategies, TO&CF incurs a large system cost than the proposed algorithm.

VI. Conclusion

In this article, we jointly investigated task offloading and service caching issues for cloud-assisted fog computing in EMS. Specifically, AIoT devices generate delay-sensitive and energy-intensive tasks. The tasks can be processed locally, and offloaded to the fog server, with the goal of minimal system cost, i.e., the weighted sum of task delay and local energy consumption. Considering dynamic service caching in the cloud-assisted fog computing framework, tasks without service support in the fog server can be further offloaded to the cloud server. To find out satisfactory solutions for task offloading and dynamic service caching, we first proposed a distributed task offloading algorithm based on noncooperative game theory, where tasks are processed in local AIoT devices or in the fog server. Then, we leveraged the 0–1 knapsack method to realize dynamic service caching based on task popularity. Guided by service caching, we adjusted offloading decisions. If tasks are offloaded to the fog server and lack computation service, the tasks will be offloaded to the cloud server or processed locally. Additionally, we conducted extensive experiments to validate the proposed algorithms, and the results show suppositories of our proposed algorithms compared with that of other algorithms. In the future, we will extend our work by adding the number of fog servers to consider migration cost in cloud-assisted fog computing networks.

References

[1] M.-Z. Pan and J.-Y. Mao, "Cross boundary mechanisms for knowledge management by user representatives in enterprise systems implementation," *IEEE Trans. Eng. Manag.*, vol. 63, no. 4, pp. 438–450, Nov. 2016.

[2] M. Aldabbas, X. Xie, B. Teufel, and S. Teufel, "Future security challenges for smart societies: Overview from technical and societal perspectives," in *Proc. Int. Conf. Smart Grid Clean Energy Technol.*, 2020, pp. 103–111.

[3] W. Zhang, S. Li, L. Liu, Z. Jia, Y. Zhang, and D. Raychaudhuri, "Hetero-edge: Orchestration of real-time vision applications on heterogeneous edge clouds," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1270–1278.

[4] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "DeepDecision: A mobile deep learning framework for edge video analytics," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 1421–1429.

[5] R. Mahmud, A. N. Toosi, K. Ramamohanarao, and R. Buyya, "Context-aware placement of industry 4.0 applications in fog computing environments," *IEEE Trans. Ind. Inform.*, vol. 16, no. 11, pp. 7004–7013, Nov. 2020.

[6] M. Mukherjee et al., "Latency-driven parallel task data offloading in fog computing networks for industrial applications," *IEEE Trans. Ind. Inform.*, vol. 16, no. 9, pp. 6050–6058, Sep. 2020.

[7] C. C. Byers, "Architectural imperatives for fog computing: Use cases, requirements, and architectural techniques for fog-enabled IoT networks," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 14–20, Aug. 2017.

[8] L. Yin, J. Luo, and H. Luo, "Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing," *IEEE Trans. Ind. Inform.*, vol. 14, no. 10, pp. 4712–4721, Oct. 2018.

[9] 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, Fourthquarter 2017.

[10] Z. Zhang, H. Zhou, and D. Li, "Joint optimization of multi-user computing offloading and service caching in mobile edge computing," in *Proc. IEEE/ACM 29th Int. Symp. Qual. Serv.*, 2021, pp. 1–2.

[11] Z. Xu et al., "To cache or not to cache: Stable service caching in mobile edge-clouds of a service market," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst.*, 2020, pp. 421–431.

[12] Y. Wu, B. Shi, L. P. Qian, F. Hou, J. Cai, and X. S. Shen, "Energy-efficient multi-task multi-access computation offloading via NOMA transmission for IoTs," *IEEE Trans. Ind. Inform.*, vol. 16, no. 7, pp. 4811–4822, Jul. 2020.

[13] 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.

[14] X. Deng, Z. Sun, D. Li, J. Luo, and S. Wan, "User-centric computation offloading for edge computing," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12559–12568, Aug. 2021.

[15] Y. Ma, W. Liang, J. Li, X. Jia, and S. Guo, "Mobility-aware and delay-sensitive service provisioning in mobile edge-cloud networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 1, pp. 196–210, Jan. 2022.

[16] R. Roshan, R. Matam, M. Mukherjee, J. Lloret, and S. Tripathy, "A secure task-offloading framework for cooperative fog computing environment," in *Proc. IEEE Glob. Commun. Conf.*, 2020, pp. 1–6.

[17] S. Bi, L. Huang, and Y.-J. 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.

[18] G. Zhao, H. Xu, Y. Zhao, C. Qiao, and L. Huang, "Offloading tasks with dependency and service caching in mobile edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 11, pp. 2777–2792, Nov. 2021.

[19] S.-W. Ko, S. J. Kim, H. Jung, and S. W. Choi, "Computation offloading and service caching for mobile edge computing under personalized service preference," *IEEE Trans. Wireless Commun.*, to be published, doi: 10.1109/TWC.2022.3151131.

[20] T. Liu, L. Fang, Y. Zhu, W. Tong, and Y. Yang, "A near-optimal approach for online task offloading and resource allocation in edge-cloud orchestrated computing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 8, pp. 2687–2700, 2022.

[21] M. Goudarzi, H. Wu, M. Palaniswami, and R. Buyya, "An application placement technique for concurrent IoT applications in edge and fog computing environments," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1298–1311, Apr. 2021.

[22] H. Wu, W. J. Knottenbelt, and K. Wolter, "An efficient application partitioning algorithm in mobile environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 7, pp. 1464–1480, Jul. 2019.

[23] X. Wang, Z. Ning, and L. Wang, "Offloading in internet of vehicles: A fog-enabled real-time traffic management system," *IEEE Trans. Ind. Inform.*, vol. 14, no. 10, pp. 4568–4578, Oct. 2018.

[24] 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.

[25] X. Dai et al., "Task co-offloading for D2D-assisted mobile edge computing in industrial Internet of Things," *IEEE Trans. Ind. Inform.*, to be published, doi: 10.1109/TII.2022.3158974.
[26] H. Jiang, Z. Xiao, Z. Li, J. Xu, F. Zeng, and D. Wang, "An energy-efficient framework for Internet of Things underlaying heterogeneous small cell networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 1, pp. 31–43, Jan. 2022.
[27] F. Zeng, Q. Li, Z. Xiao, V. Havyarimana, and J. Bai, "A price-based optimization strategy of power control and resource allocation in full-duplex heterogeneous macrocell-femtocell networks," *IEEE Access*, vol. 6, pp. 42004–42013, 2018.
[28] Z. Xiao et al., "A joint information and energy cooperation framework for CR-Enabled macro-femto heterogeneous networks," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2828–2839, Jul. 2020.
[29] H. Jiang, X. Dai, Z. Xiao, and A. K. Iyengar, "Joint task offloading and resource allocation for energy-constrained mobile edge computing," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2022.3150432.
[30] X. Hu, K.-K. Wong, K. Yang, and Z. Zheng, "UAV-assisted relaying and edge computing: Scheduling and trajectory optimization," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4738–4752, Oct. 2019.
[31] 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.
[32] S. Hart and A. Mas-Colell, "A simple adaptive procedure leading to correlated equilibrium," *Econometrica*, vol. 68, no. 5, pp. 1127–1150, 2010.
[33] Z. Xiao et al., "Spectrum resource sharing in heterogeneous vehicular networks: A noncooperative game-theoretic approach with correlated equilibrium," *IEEE Trans. Veh. Technol.*, vol. 67, no. 10, pp. 9449–9458, Oct. 2018.
[34] Z. Xiao et al., "Vehicular task offloading via heat-aware MEC cooperation using game-theoretic method," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 2038–2052, Mar. 2020.
[35] V. Farhadi et al., "Service placement and request scheduling for data-intensive applications in edge clouds," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1279–1287.
[36] R. Jindal, N. Kumar, and H. Nirwan, "TFCT: A task offloading approach for fog computing and cloud computing," in *Proc. 10th Int. Conf. Cloud Comput. Data Sci. Eng.*, 2020, pp. 145–149.
[37] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "Stackelberg game for service deployment of IoT-enabled applications in 6G-Aware fog networks," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5185–5193, Apr. 2021.

Xingxia Dai (Student Member, IEEE) received the B.S. degree in communication engineering from Xiangtan University, Xiangtan, China, in 2018. She is currently working toward the Ph.D. degree in computer science and technology with Hunan University, Changsha, China.

Her current research interests include internet of vehicles and mobile edge computing.

Zhu Xiao (Senior Member, IEEE) received the M.S. and Ph.D. degrees in communication and information system from Xidian University, Xi'an, China, in 2007 and 2009, respectively.

From 2010 to 2012, he was a Research Fellow with the Department of Computer Science and Technology, University of Bedfordshire, Luton, U.K. He is currently an Associate Professor with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. His research interests include mobile communications, wireless localization, Internet of Vehicles, and trajectory data mining.

Hongbo Jiang (Senior Member, IEEE) received the Ph.D. degree in computer science from Case Western Reserve University, Cleveland, OH, USA, in 2008.

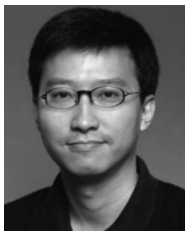
He is currently a Full Professor with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. He was a Professor with the Huazhong University of Science and Technology, Wuhan, China. His research interests include computer networking, especially algorithms and protocols for wireless and mobile networks.

Dr. Jiang was the Editor for IEEE/ACM TRANSACTIONS ON NETWORKING, the Associate Editor for IEEE/ACM TRANSACTIONS ON MOBILE COMPUTING, and the Associate Technical Editor for *IEEE Communications* magazine. He is an elected member of Academia Europaea, Fellow of Institution of Engineering and Technology (IET), Fellow of The British Computer Society (BCS), and Fellow of The Asia-Pacific Artificial Intelligence Association (AAIA).

Mamoun Alazab received the Ph.D. degree in computer science from the School of Science, Information Technology and Engineering, Federation University Australia, Ballarat, Australia, in 2012.

He is currently an Associate Professor with the College of Engineering, IT and Environment, Charles Darwin University, Darwin, Australia. He is a Cyber Security Researcher and a Practitioner with industry and academic experience. He has authored or coauthored more than 200 research papers in many international journals and conferences. His research is multidisciplinary that focuses on cyber security and digital forensics of computer systems with a focus on cybercrime detection and prevention.

Dr. Alazab is the Founding Chair of the IEEE Northern Territory (NT) Subsection.



John C. S. Lui (Fellow, IEEE) was born in Hong Kong. He received the Ph.D. degree in computer science from the University of California, Los Angeles, CA, USA, in 1992.

He is currently the Choh-Ming Li Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong (CUHK), Hong Kong, China, where he was the Chairman of the department from 2005 to 2011. His current research interests include communication networks, network/system security (e.g., cloud security, mobile security, etc.), network economics, network sciences (e.g., online social networks, information spreading, etc.), cloud computing, large-scale distributed systems, and performance evaluation theory.

Dr. Lui is an elected member of the International Federation for Information Processing Working Group (IFIP WG) 7.3, a Fellow of the Association for Computing Machinery (ACM), a Senior Research Fellow of the Croucher Foundation, and was the Chair of the ACM SIGMETRICS. He has been serving in the Editorial Board of the IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, *Performance Evaluation*, and the *International Journal of Network Security*. He is the recipient of various departmental teaching awards and the CUHK Vice-Chancellors Exemplary Teaching Award. He is also a corecipient of the Best Paper Award in the IFIP WG 7.3 Performance 2005, IEEE/IFIP Network Operations and Management Symposium (NOMS) 2006, and SIMPLEX 2013.

Geyong Min (Member, IEEE) received the B.Sc. degree from the Huazhong University of Science and Technology, Wuhan, China, and the Ph.D. degree from the University of Glasgow, Glasgow, Scotland, U.K., in 1995 and 2003, respectively, both in computer science.

He is a Professor of high performance computing and networking with the Department of Computer Science within the College of Engineering, Mathematics, and Physical Sciences, University of Exeter, Exeter, U.K. His research interests include future internet, computer networks, wireless communications, multimedia systems, information security, high performance computing, ubiquitous computing, modeling and performance engineering.

Schahram Dustdar (Fellow, IEEE) received the Ph.D. degree in business informatics from the University of Linz, Linz, Austria, in 1992.

He is currently a Full Professor of computer science (informatics) with a focus on internet technologies heading the Distributed Systems Group, TU Wien, Wein, Austria. Since December 2016, he has been the Chairman of the Informatics Section of the Academia Europaea.

Dr. Dustdar has been a Member of the IEEE Conference Activities Committee (CAC), since 2016, the Section Committee of Informatics of the Academia Europaea, since 2015, and the Academia Europaea: The Academy of Europe, Informatics Section, since 2013. He was a recipient of the Association for Computing Machinery (ACM) Distinguished Scientist Award in 2009 and the International Business Machines Corporation (IBM) Faculty Award in 2012. He is an Associate Editor for IEEE TRANSACTIONS ON SERVICES COMPUTING, *ACM Transactions on the Web*, and *ACM Transactions on Internet Technology*. He is on the Editorial Board of IEEE.

Jiangchuan Liu (Fellow, IEEE) received the B.Eng. degree (cum laude) from Tsinghua University, Beijing, China, and the Ph.D. degree from the Hong Kong University of Science and Technology, Hong Kong, China, in 1999 and 2003, respectively, both in computer science.

He is a University Professor with the School of Computing Science, Simon Fraser University, British Columbia, Canada. He was as an Assistant Professor with The Chinese University of Hong Kong, Hong Kong, China, and a Research Fellow with Microsoft Research Asia. His research interests include multimedia systems and networks, cloud and edge computing, social networking, online gaming, and Internet of Things/RFID/backscatter.

Dr. Liu is a Fellow of The Canadian Academy of Engineering and a Natural Sciences and Engineering Research Council of Canada (NSERC) E.W.R. Steacie Memorial Fellow. He was an EMC Endowed Visiting Chair Professor of Tsinghua University (2013–2016). He is a corecipient of the inaugural Test of Time Paper Award of IEEE International Conference on Computer Communications (INFOCOM) (2015), Association for Computing Machinery (ACM) Special Interest Group on Multimedia (SIGMM) Transaction on Multimedia Computing, Communication and Applications (TOMCCAP) Nicolas D. Georganas Best Paper Award (2013), and ACM Multimedia Best Paper Award (2012). He has served on the editorial boards of IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON BIG DATA, IEEE TRANSACTIONS ON MULTIMEDIA, *IEEE Communications Surveys and Tutorials*, and *IEEE Internet of Things* journal. He is a Steering Committee member of IEEE TRANSACTIONS ON MOBILE COMPUTING and Steering Committee Chair of *IEEE/ACM International Symposium on Quality of Service (*IWQoS) (2015–2017). He is TPC Cochair of IEEE International Conference on Computer Communications (INFOCOM)'2021.