Joint Optimization of Sensing and Computation for Status Update in Mobile Edge Computing Systems

Yi Chen, Zheng Chang, *Senior Member, IEEE*, Geyong Min, *Senior Member, IEEE*, Shiwen Mao, *Fellow, IEEE*, and Timo Hämäläinen, *Senior Member, IEEE*

***Abstract*— IoT devices have been widely utilized to detect state transition in the surrounding environment and transmit status updates to the base station for system operations. To guarantee the accuracy of system control, age of information (AoI) is introduced to quantify the freshness of the sensory data and meet the stringent timeliness requirement. Due to the limited computing resources, the status update can be offloaded to the mobile edge computing (MEC) server for execution. Since status updates generated by insufficient sensing operations may be invalid and lead to additional processing time, a joint data sensing and processing optimization problem needs to be considered. Therefore, this work formulates an NP-hard problem that considers the freshness of the status updates and energy consumption of the IoT devices. Subsequently, the problem is decomposed into sampling, sensing, and computation offloading optimization problems. To optimize the system overhead, a multi-variable iterative system cost minimization algorithm is proposed. Simulation results illustrate the efficacy of our method in decreasing the system cost, and indicate the influence of sensing and processing under different scenarios.**

***Index Terms*— Age of information, mobile edge computing, computation offloading, status update.**

I. Introduction

A. Background and Motivation

With the development of Internet of Things (IoT), ubiquitous connections of billions of IoT devices ranging from tiny IoT sensors to more powerful smarts phones are envisioned [1]. To realize various IoT applications like object recognition, traffic monitoring and autonomous driving, vast information from the physical world should be extracted and transformed into status updates to implement intelligent control for IoT devices [2]. As the processing operations of the status update are time-consuming and computation-intensive, it is an arduous task to process the sensory data timely for IoT devices with limited storage and computation capacity. Mobile edge computing (MEC) has been regarded as a promising technology to overcome the resource constraints of IoT devices by providing cloud-like computing resources at the network edge [3], [4]. In this case, IoT devices are able to offload the computing tasks to the nearby MEC server for further execution. By deploying powerful computing resources in proximity to IoT devices and executing computing tasks on behalf of IoT devices, there is a significant potential to boost the development of IoT with extensive applications [5], [6]. For example, the implementation of MEC-assisted IoT systems has led to the emergence of a variety of real-time sensing applications, such as healthcare monitoring [7], intelligent agriculture [8], and disaster detection [9], that have greatly benefited from this technology. In the context of smart agriculture, the MEC-assisted IoT system offers a more efficient method for collecting and analyzing data from a field or crop. By deploying various IoT devices to gather information on factors such as soil, weather, and irrigation and processing status information with the help of edge servers, the system is able to quickly and effectively process this data, making it available to a central control unit for decision-making purposes. This allows for more informed choices on when to plow the field and when to harvest the crop.

Moreover, the accurate monitoring of the IoT system relies on the strict requirement on the freshness of the collected information. Recently, age of information (AoI) defined as the time elapsed since the generation of the last status update, has been adopted as a performance metric to quantify the freshness of generated status information [10]. When an IoT device generates and receives a status update successfully, its value of AoI is reset to 0. The AoI of the status update increases linearly with time until the next status update is successfully received. The average value of the AoI during the continuous sensing periods reflects the freshness of the sensory data [11]. By introducing the concept of AoI, the abstract information freshness problem can be transformed into a concrete mathematical optimization problem.

Manuscript received 11 November 2022; revised 20 February 2023; accepted 11 March 2023. Date of publication 30 March 2023; date of current version 13 November 2023. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62071105, in part by the Sichuan Natural Science Foundation under Grant 2022NSFSC0544, and in part by the NSF under Grant ECCS-1923717. The associate editor coordinating the review of this article and approving it for publication was D. Niyato. *(Corresponding author: Zheng Chang.)*

Yi Chen is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China.

Zheng Chang is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China, and also with the Faculty of Information Technology, University of Jyväskylä, 40014 Jyväskylä, Finland (e-mail: zheng.chang@jyu.fi).

Geyong Min is with the Department of Computer Science, University of Exeter, EX4 4QF Exeter, U.K.

Shiwen Mao is with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849 USA.

Timo Hämäläinen is with the Faculty of Information Technology, University of Jyväskylä, 40014 Jyväskylä, Finland.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TWC.2023.3261338.

Digital Object Identifier 10.1109/TWC.2023.3261338

The typical IoT system can be constructed as a three-layered structure including the sensing layer, the network layer and the application layer [12]. Sensing of the IoT device is considered as the process that the IoT device observes the environment transition and generates the status update when necessary. In the sensing layer, the IoT devices keep sensing the state transition process and generating state updates periodically. The periodic sensing model is the preferred choice for applications requiring constant monitoring of crucial conditions or processes, including temperature, pressure, and humidity [13], [14], [15]. This model ensures real-time surveillance, promoting consistent and accurate data collection. Then, the sensory data are transmitted through the network layer for processing and future system control. Processing of the status update includes determining whether a status update is valid and extracting the necessary information from the status update to perform system control. By repeatedly receiving the valid status update from different IoT devices, the AoI at the BS can be reduced during the system control process [16]. Although numerous works have been done on how to reduce the AoI in IoT system, prior works mainly focus on the joint optimization of sensing and communication [2], [16], [28]. However, data processing, which also plays a crucial role in determining the freshness of the data, needs to be considered concurrently with the sensing part due to the connection between the sensing time and the processing time.

In the sensing layer, in order to ensure the information freshness on the BS side, IoT devices should perform the sampling operations and generate status updates at as high a frequency as possible. With the short sampling interval, the BS can achieve a low value of AoI due to the frequent status updates [17]. But sampling the state transition frequently brings additional energy consumption, which is not negligible for IoT devices with limited battery capacity. Moreover, successful sensing may be a random event for IoT devices due to the possible state tracking error rate [18], [19]. Longer sensing time can significantly improve the sensing successful probability, at the cost of additional time overhead and freshness of status updates. In addition, if the sensing time is insufficient, multiple sensing failures can lead to multiple unnecessary repetitions of the sensing process. As the data sensing-processing procedure shown in Fig. 1, the IoT device that performs sensing operations three times will receive a longer sensing time and a short total task duration time. Nevertheless, the IoT device which only performs a single sensing operation generates the invalid update unfortunately and has to execute the extra sensing-processing procedure until the sensing operation is successful. Thus, the sensing time of the IoT device should be explicitly considered to obtain the minimum time overhead.

In the processing part, since some information embedded in the sensory data requires further processing before it can be utilized for further system control, the execution time of the sensing task of the IoT device also has a significant impact on the freshness of the status update. Due to the computation capacity limitations of the IoT devices themselves, they would prefer to offload the tasks to the MEC server for execution to obtain a shorter processing time. But selfish IoT devices will compete for limited computational and communication resources, which may create utility conflicts and increase the system overhead during the computation offloading decision-making process [20], [21].

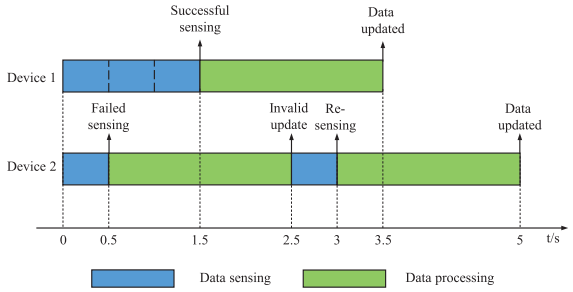

Fig. 1. Example of the sensing-processing procedure.

B. Contribution

In view of these above-mentioned challenges, we further analyze the joint optimization problem of sensing and processing in MEC system. By investigating the effects of sampling, sensing, and processing on system overhead independently, we seek to realize the IoT device sensing and processing trade-off in terms of the information freshness and energy consumption for IoT devices. The main contributions of the paper can be summarized as follows.

- We first introduce an MEC-assisted IoT system, where the IoT devices keep generating status updates periodically. Then, the sensing, data transmission and task processing are modeled separately and the AoI of the sensory data during the status updating process is formulated.
- A joint sensing and processing optimization problem is formulated for status updates to minimize the system overhead including the energy consumption and the information freshness. Then, the NP-hard problem is transformed into three subproblems to optimize the sampling interval, sensing time and computation offloading decision individually.
- We solve the sensing and sampling subproblems with extremum principles, and solve the computation offloading decision-making problem with a game-theoretic approach. Then, a multi-variable iteration system cost optimization algorithm (MISCO) is proposed to minimize the system overhead.
- Extensive simulations are conducted to present the effectiveness of our proposed optimization algorithm. Numerical results illustrate the connection between data sensing and processing.

C. Organization

The rest of this paper is organized as follows. In Section II, a literature review is conducted. In Section III, we present the MEC-assist IoT system model and formulate the sensing, transmission and processing models during the status update process. In Section IV, we analyze the AoI in the considered system. In Section V, we propose the system overhead optimization problem and decompose the problem into sensing,

sampling and processing subproblems. In Section VI, the joint sensing and processing optimization algorithm MISCO is proposed. In Section VII, simulation results are presented and discussed. Finally in Section VIII, we conclude our paper.

II. Related Work

Freshness of the status update has emerged as a recent highlight in the field of network research, which leads to the increasing research interest in AoI served as the metrics to measure the freshness of information [22], [23], [24], [25], [26], [27]. Yates et al. in [22] investigate real-time status updates generated by multiple independent sources sending to a single monitor with an AoI timeliness metric and derive the general values of AoI suitable for various multi-source service system. Kadota et al. in [24] study a single-hop wireless network where multiple nodes transmit time-sensitive information to the base station while minimizing the expected weighted total AoI of the network and satisfying the just-in-time throughput at the same time. Feng et al. in [25] design an optimal strategy for the energy harvesting sensor to generate status updates with the purpose of minimizing the long-term average AoI and satisfying the energy constraint in the different cases of whether the system has the updating feedback. Zhou et al. in [26] study a time-intensive IoT monitoring system where IoT devices continuously generate and transmit the status updates with updating cost. Through simultaneously optimizing the sampling and updating process, the minimum average AoI of the destination node is derived under the upper bound of the updating cost. Chen et al. in [27] investigate the AoI-aware radio resource management problem in a Manhattan Grid vehicle-to-vehicle network to realize the optimal frequency allocation and packet scheduling decision-making.

Several relative works have been conducted in the context of the optimization of data sensing [2], [16], [28], [29], [30]. Peng et al. in [2] propose a joint sensing and communication scheduling framework for status updates in the multi-access network to minimize the average status error. The trade-off between data sensing and data transmission is investigated to minimize the long-term average AoI during multiple sensing cycles in [16] and [28]. In [29] and [30], the UAV trajectory optimization problem where UAV performs the sensing tasks to collect the time-intensive data is studied to satisfy the system AoI threshold.

In addition to the issues mentioned above, the collected data should be offloaded to the MEC server for low-latency processing. Due to the resource competition among IoT devices, the computation offloading optimization problem needs to be considered. The computation offloading decision-making problem has been widely investigated to address the computation capacity constraint and communication interference. Game-theoretic method has been introduced to address the computation offloading decision-making problem [31], [32], [33], [34]. Chen et al. in [31] first utilize the concept of the potential game to achieve the Nash equilibrium of the computation offloading game. Yang et al. in [32] propose a computation offloading game including multiple computation offloading schemes which take advantage of the available resources of idle mobile devices.

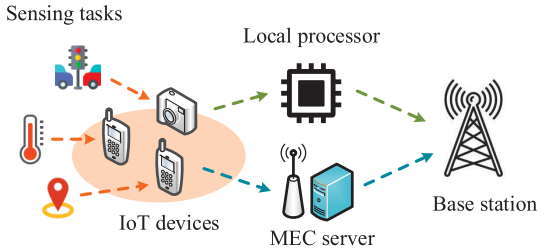

Fig. 2. Example of MEC-assisted IoT system.

However, most of the studies have focused on reducing processing latency and system energy consumption while performing computation offloading decision-making, ignoring the role of information freshness in improving service quality. In addition, the state-sensing procedure may generate invalid state updates, resulting in extra processing time and deteriorating the service quality. Therefore, the sensing and processing operations need to be considered jointly to ensure the freshness of the status update and minimize the system overhead.

III. System Model

A. Network Model

We consider a typical MEC-assisted IoT system with a set $\mathcal{N}$ of N IoT devices and an MEC server. The IoT devices monitor a physical process like the traffic condition in the autonomous driving system. The IoT devices sample the status information and generate the status update when necessary. The status sampling process of each IoT device i is independent of each other and takes the periodic delivery sampling policy of sensor updates which is one of the most common approaches in practice [36]. The sampling intervals of IoT devices are denoted as $\mathrm{T} = \{\tau_1, \tau_2, \ldots, \tau_N\}$. The MEC server functions as an MEC server provider which is located in proximity to the IoT devices and can be accessed by the IoT devices via the wireless channel. IoT devices that transmit computing tasks to the MEC server will be associated with clones at the MEC server which execute computing tasks on behalf of IoT devices [35]. Considering the constraint of IoT devices' computation capability, each IoT device can choose to process the status information locally with its own processor or to offload the sensed data to the MEC server or more powerful computing resources. After extracting the required information from the raw status data, the computing results are transmitted to the BS for future system control. To keep the freshness of the status information and guarantee accurate control, the sensing, transmission and processing process needs to be executed repeatedly. Fig. 2 shows the status offloading and processing procedure.

B. Sensing Model

In this section, we describe the data sensing process of generating status updates for IoT devices. Let t_i^{unit} be the sensing time for IoT device i to perform a whole sensing task once. The status update processing task generated by IoT

device i can be represented as a tuple $U_i = \{d_i, c_i\}$, where d_i denotes the size of the sensory data generated in one sensing operation and c_i denotes the necessary CPU cycles to finish the computing tasks.

To evaluate the sensing quality of IoT devices, we utilize a probabilistic sensing model [37]. When an IoT device executes a sensing task, the successful sensing possibility is denoted as

$$\varrho_i = e^{-\xi D_i^s}, \tag{1}$$

where D_i^s is the distance between the IoT device and the sensing target, and ξ is a positive parameter to evaluate the quality of IoT device detection depending on the environmental condition. Considering one single data sensing operation cannot satisfy the successful possibility requirement, the device may repeat the sensing operation to improve the sensing successful possibility. Let $S = \{s_1, s_2, \ldots, s_N\}$ be the number of sensing operations executed by the IoT devices in a sensing operation. After finishing multiple sensing operations, the sensing successful possibility is denoted as

$$P_i(s_i) = 1 - (1 - \varrho_i)^{s_i}. \tag{2}$$

To ensure the sensing quality, the successful sensing possibility should be lower bounded. Let p_{min} be the threshold for the successful sensing possibility of IoT devices. When IoT devices execute sensing operations, the successful sensing probability should satisfy:

$$P_i(s_i) \geq p_{min}, \quad \forall i \in \mathcal{N}. \tag{3}$$

Note that as the number of sensing operations increases, the IoT device achieves a higher sensing successful possibility. However, the multiple sensing operations will lead to a longer sensing time and larger sensing energy consumption. For device i, the sensing time is

$$T_i^{\text{ses}}(s_i) = t_i^{\text{unit}} s_i. \tag{4}$$

To process the generated status update in time, the sensing time should not exceed the sampling frequency. Otherwise, before the information of status update is extracted, another new status update is generated by the IoT device, which makes the information of the former status update outdated. Thus, the sensing time should satisfy the constraint:

$$1 \leq s_i \leq \left\lfloor \frac{\tau_i}{t_i^{\text{unit}}} \right\rfloor. \tag{5}$$

Let e_i be the energy cost for IoT device i to detect the status information per bit sensed data, the energy consumption of one single sensing process is

$$E_i^{\text{ses}}(s_i) = e_i d_i s_i. \tag{6}$$

It is worth noticing that the IoT devices cannot know whether the status update is generating successfully from the raw sensory data. Further processing operations are required to verify the validity of the generated status update. If the sensing process is failed, the sensory data will be removed and the control unit will send the request to generate another new status update to the IoT device. If the IoT device has generated a new status update before receiving the request, the request will not be further processed. Otherwise, the IoT device will restart the sensing process immediately regardless of the sampling interval.

C. Transmission Model

When finishing the sensing procedure and generating a new status update, the IoT devices need to further process the status update to verify the validity of the status update and extract the status information. Since some IoT devices are limited in computation capability, they need to transmit their status updates to the edge server for further processing. Let $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$ be the transmission policies for all the N IoT devices, in which the elements can be expressed as follows.

$$x_i = \begin{cases} 1, & \text{IoT device } i \text{ transmits to edge server.} \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

When the IoT device decides to transmit its status update to the edge server for processing, the transmission rate for the status update can be written as

$$r_i(\mathbf{x}) = B \log_2 \left(1 + \frac{g_{i,s} p_i}{\omega_0 + \sum_{m \in \mathcal{N}, m \neq i} x_m g_{m,s} p_m} \right), \tag{8}$$

where B is the channel bandwidth allocated to the IoT device i, g_i is the channel gain between device i and the edge node, p_i represents the device i's transmission power, and ω_0 represents the noise power. Then, the transmission latency is

$$T_i^{\text{trans}}(\mathbf{x}) = \frac{d_i}{r_i(\mathbf{x})}. \tag{9}$$

and the transmission energy consumption can be expressed by

$$E_i^{\text{trans}}(\mathbf{x}) = \frac{p_i d_i}{r_i(\mathbf{x})}. \tag{10}$$

D. Computation Model

In this subsection, we introduce the computation model of the status update. Dependent on the different computation offloading strategies of the IoT devices, the time and energy to execute the computation tasks of the status updates are different. If the device chooses to process the status update with its own local processor, the computation latency is $T_i^{\text{local}} = \frac{c_i}{f_i}$, where f_i represents the CPU frequency of the local processor in IoT device i. Besides, processing the computation task locally brings extra energy consumption to the IoT device itself, which is calculated as

$$E_i^{\text{local}} = c_i \delta. \tag{11}$$

where δ is the energy consumption cost per CPU cycle. When the computation task is transmitted to the edge server for further processing, the computation latency is expressed as

$$T_i^{\text{edge}} = \frac{c_i}{f^e}. \tag{12}$$

where f^e denotes the CPU frequency of the edge server.

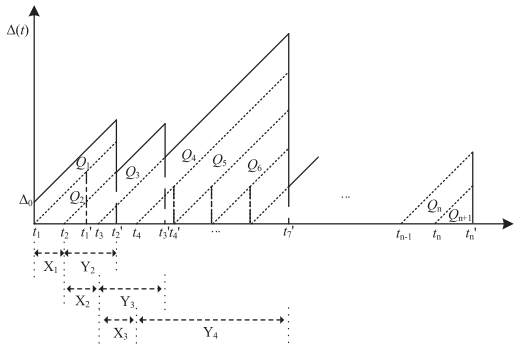

Fig. 3. Example of AoI.

IV. Age of Information Analysis

We introduce the concept of AoI to evaluate the freshness of the status update generated by IoT devices. Let $A_i(t)$ be the AoI of the IoT device i with $A_i(0) = 0$, and the AoI is defined as the time that elapses from the last valid status update received by the control unit. One status update is valid to the control unit when the sensing process of status update is successful and the computation of the status update is finished. The AoI of status update grows from the beginning of the sensing, and keeps increasing until the next valid status update is received by the control unit. Let $T_i^{j,\text{prcs}}$ be the processing time to conduct processing operation j, and $T_i^{j,\text{prcs}} = j \times T_i^{1,\text{prcs}}$. Based on the different offloading strategies, the processing time of one single process operation is calculated as

$$T_i^{1,\text{prcs}}(s_i, \mathbf{x}) = T_i^{\text{ses}}(s_i) + \left(T_i^{\text{trans}}(\mathbf{x}) + T_i^{\text{edge}}\right) x_i \quad (13)$$

$$+ T_i^{\text{local}}(1 - x_i). \quad (14)$$

However, the sensing time might be too short for IoT devices to successfully generate a valid status update with only one sensing operation. After finishing the computation execution, the control unit may find the uploaded status update fails to meet the requirement, i.e. the sensing processing is not successful. In this case, the status update will be dismissed and a request will be sent to the IoT device to repeat the sensing process to generate another status update. Then we have the following **Theorem 1** on the average number of processing time.

Theorem 1: For IoT device i, the average number of processing time for IoT devices to finish a processing task is $\frac{T_i^{1,prcs}(s_i,\mathbf{x})}{P_i}$.

Proof: Please see Appendix A.

Next, we derive the AoI of the status updates. Note that the information embedded in the status update can be used for further execution after processing. The AoI at time t of the i-th IoT device is denoted as

$$\Delta_i(t) = t - a_i(t), \quad (15)$$

where $a_i(t)$ is the time when the latest status update generated by the IoT device i is successfully sensed and processed. Without loss of generality, we assume the initial observing time is $t_1 = 0$, and the initial AoI is Δ_0. As shown in Fig. 3, the AoI grows linearly during the sensing and processing procedure and reset to a smaller value when a new status update is successfully accepted. Let t_j be the time when the j-th status update is generated and finishes processing at the time t'_j. Let Y_j denote the system time of the status update j, which is defined as

$$Y_j = t'_u - t_j, \quad (16)$$

where t'_u is the time when the next status update is successfully sensed and executed. Specifically, when the status update j is valid, $Y_j = t'_j$, i.e. the finishing time of the status update j. Besides, X_j denotes the time between the generation of two continuous status update j and $j+1$, which is given by

$$X_j = t_{j+1} - t_j. \quad (17)$$

Based on the definition above, the average AoI of the IoT device i is denoted as

$$\overline{\Delta_i} = \frac{1}{T}\int_0^T \Delta_i(t)\,\mathrm{d}t. \quad (18)$$

For the sake of simplicity, we consider the time interval with $T = t'_n$ which is displayed in Fig. 3. To calculate the average AoI, the area is divided into several geometric parts which are expressed by the concatenation of polygons Q_j. Hence, the average AoI is given by

$$\overline{\Delta_i} = \frac{1}{T}\sum_{j=1}^{n+1} Q_j. \quad (19)$$

The area of the polygons is calculated differently. For $j = 1$, $Q_1 = \Delta_0(X_1 + Y_2)$. For $2 \le j \le n$, the area of the trapezoid Q_j is calculated by the difference between two triangles, which is

$$Q_j = \frac{1}{2}(X_{j-1} + Y_j)^2 - \frac{1}{2}Y_j^2 = \frac{1}{2}X_{j-1}^2 + Y_j X_{j-1}. \quad (20)$$

Besides, the area of the Q_{n+1} is the area of a triangle with a width of Y_n. Hence, the equation (19) can be rewritten as

$$\overline{\Delta_i} = \lim_{T\to\infty} \frac{Q_1 + Q_{n+1} + \sum_{j=2}^{n} Q_j}{T} = \lim_{T\to\infty}\left[\frac{Q_1+Q_{n+1}}{T} + \frac{n-1}{T}\frac{\sum_{j=2}^{n}(\frac{1}{2}X_{j-1}^2 + Y_j X_{j-1})}{n-1}\right]. \quad (21)$$

As T becomes larger, i.e. $T \to \infty$, the value of $\frac{Q_1+Q_{n+1}}{T}$ can be ignored consequently and $\frac{n-1}{T}$ can be treated as the value of $\frac{1}{\mathbb{E}[X]}$. From the analysis above, we have

$$\overline{\Delta_i} = \frac{\mathbb{E}[Q]}{\mathbb{E}[X]} = \frac{\frac{1}{2}\mathbb{E}[X^2] + \mathbb{E}[XY]}{\mathbb{E}[X]}. \quad (22)$$

Considering the sensing model mentioned above, the value of $\mathbb{E}[X]$ is dependent on the sampling interval, which is given by $\mathbb{E}[X] = \tau_i$.

Besides, the value of $\mathbb{E}[Y]$ is identical to the average processing time of a successful status update $\mathbb{E}[T_i^{\text{prcs}}]$. Since X_j is independent of Y_j, we derive the average AoI as

$$\overline{\Delta_i}(s_i, \tau_i, \mathbf{x}) = \frac{1}{2}\mathbb{E}[X] + \mathbb{E}[Y] = \frac{1}{2}\tau_i + \frac{T_i^{1,\text{prcs}}(s_i, \mathbf{x})}{P_i(s_i)}. \quad (23)$$

V. Problem Formulation

A. Problem Formulation

The freshness of the generated status update plays a key role in accurate monitoring and controlling. Hence, the AoI should be well considered when evaluating the system performance. Besides, energy consumption is another significant performance metric due to the physical constraint of IoT devices. The sensing, transmission and computation operations all consume considerable energy during the running procedure of IoT devices. The average value of IoT device i is dependent on the energy consumption per time slot, which is given by

$$\begin{aligned}\overline{E_i}\left(s_i,\tau_i,\mathbf{x}\right) &= \lim_{T\to\infty}\frac{\sum_{j=1}^{n}\frac{E_i^{\text{ses}}(s_i)+E_i^{\text{tran}}(\mathbf{x})x_i+E_i^{\text{local}}(1-x_i)}{P_i}}{T}\\ &= \frac{E_i^{\text{ses}}(s_i)+E_i^{\text{tran}}(\mathbf{x})x_i+E_i^{\text{local}}(1-x_i)}{\mathbb{E}[X]P_i}\\ &= \frac{E_i^{\text{ses}}(s_i)+E_i^{\text{tran}}(\mathbf{x})x_i+E_i^{\text{local}}(1-x_i)}{P_i\tau_i}. \quad (24)\end{aligned}$$

As discussed above, the overhead of the computation offloading problem includes both the average AoI and energy consumption, which can be formulated as

$$C_i(s_i,\tau_i,\mathbf{x}) = \mu_t\overline{\Delta_i}\left(s_i,\tau_i,\mathbf{x}\right)+\mu_e\overline{E_i}\left(s_i,\tau_i,\mathbf{x}\right), \quad (25)$$

where μ_t and μ_e are the scalar weight of the AoI and energy consumption respectively to measure the both fairly and control the trade-off between the AoI and energy consumption. For all the IoT devices, the optimization objective is to minimize the total overhead, which is expressed by

$$\begin{aligned}\mathcal{P}: \min_{S,\tau,\mathbf{x}} \ & \sum_{i=1}^{N} C_i(s_i,\tau_i,\mathbf{x}) && (26a)\\ s.t.\ & 1\le s_i\le\left\lfloor\frac{\tau_i}{t_i^{\text{unit}}}\right\rfloor \quad \forall i\in\mathcal{N}, && (26b)\\ & x_i\in\{0,1\}, \quad \forall i\in\mathcal{N} && (26c)\\ & \tau_i\ge\tau_{min}, \quad \forall i\in\mathcal{N} && (26d)\\ & P_i\left(s_i\right)\ge p_{min}, \quad \forall i\in\mathcal{N} && (26e)\\ & \sum_{i=1}^{N} x_i d_i\le D_e. \quad \forall i\in\mathcal{N} && (26f)\end{aligned}$$

where D_e is the data threshold of the MEC server. Constraints in (26b) ensure the sensing time for a IoT device will not exceed the sampling interval. (26c) guarantees the offloading decision for each IoT device is binary. (26d) is the lower bound of the sampling interval for IoT devices. (26e) is the lower bound of the successful sensing probability. Constraint in (26f) means the upper bound of the data size of the MEC server.

B. Problem Decomposition

Considering $\mathbf{s}$ and $\mathbf{x}$ are both discrete variables, the feasible set of Problem (26a) is non-convex. Besides, the variables contain both continuous variables and discrete variables, which makes the optimization problem NP-hard [38]. In this part, we decompose the optimization problem into several subproblems: sampling interval optimization, sensing optimization and computation offloading optimization.

1) Sampling Interval Optimization: Due to the constraint (26b), the upper bound of the sensing time is dependent on the sampling interval of IoT devices. Hence, to obtain the optimal sensing time, the sampling period should be determined first. Note that the sampling interval has a great influence on the AoI and energy consumption of IoT devices. When the IoT devices generate status updates more frequently, i.e. the smaller τ_i for IoT device i, the AoI decreases accordingly. However, the energy consumption will increase greatly due to the frequent sampling action. In this subproblem, we study the optimal sampling interval for IoT devices to achieve the trade-off between the AoI and energy consumption, which is denoted as

$$\mathcal{P}_1: \ \min_{\mathrm{T}}\sum_{i=1}^{N} C_i(\tau_i) \quad s.t.\ (26d). \quad (27)$$

2) Sensing Time Optimization: Based on the result of the sampling interval optimization, the upper bound of sensing time is obtained. With more sensing times, the sensing successful probability is greatly improved. However, the excessive sensing operation may lead to unnecessary sensing latency and extra sensing energy consumption. To determine the suitable sensing time for IoT devices, the problem $\mathcal{P}$ is rewritten as

$$\mathcal{P}_2: \ \min_{S}\sum_{i=1}^{N} C_i(s_i) \quad s.t.\ (26b),(26e). \quad (28)$$

3) Computation Offloading Optimization: After solving $\mathcal{P}_1$ and $\mathcal{P}_2$, our goal is to find an optimal computation offloading policy for all the IoT devices to minimize the system overhead. The problem can be expressed as

$$\mathcal{P}_3: \ \min_{\mathbf{x}}\sum_{i=1}^{N} C_i(\mathbf{x}) \quad s.t.\ (26c),(26f). \quad (29)$$

Although the MEC server is equipped with powerful computation capability, more IoT devices choosing to transmit computing tasks to MEC server will cause severe interference which may lead to extra time latency. Based on these observations, we aim to optimize the computation offloading strategies for IoT devices to minimize the system overhead.

VI. Joint Optimization of Sensing and Computation

The joint optimization of sensing and computation is realized using the Block Coordinate Descent (BCD) method. The fundamental concept of BCD involves alternating between optimizing one variable while holding the others fixed until the objective function reaches convergence. By fixing the sampling interval, sensing time, and computation offloading policy separately, we solve the subproblems proposed above. Then, we design an iterative algorithm to solve the problem $\mathcal{P}$ to minimize the system overhead jointly.

A. Sampling Interval Optimization

In this part, we solve the sampling interval optimization problem $\mathcal{P}_1$ mentioned in (27). Given the fixed sensing time and computation offloading policy, the value of $\frac{T_i^{1,\text{prcs}}}{P_i}$ remains

unchanged. Therefore, for simplicity, the expression of $\mathcal{P}_1$ can be rewritten as

$$\mathcal{P}_1: \min_{\mathrm{T}} \sum_{i=1}^{N} C_i^t(\tau_i) = \sum_{i=1}^{N} \left(\frac{\mu_t \tau_i}{2} + \mu_e \frac{E_i^{\mathrm{ses}}(s_i) + E_i^{\mathrm{tran}}(\mathbf{x}) x_i + E_i^{\mathrm{local}}(1-x_i)}{P_i \tau_i} \right)$$
$$s.t.\ (26d). \tag{30}$$

Since τ_i is the continuous variable, we calculate the derivative directly to discuss the variation trend to address the optimization problem. Then, the derivative of $C_i^t(\tau_i)$ is calculated as

$$\frac{\partial C_i^t(\tau_i)}{\partial \tau_i} = \frac{\mu_t}{2} - \frac{\mu_e E_i^{\mathrm{total}}}{\tau_i^2}, \tag{31}$$

where $E_i^{\mathrm{total}} = \frac{E_i^{\mathrm{ses}}(s_i) + E_i^{\mathrm{tran}}(\mathbf{x}) x_i + E_i^{\mathrm{local}}(1-x_i)}{P_i}$ is a constant.

By solving $\frac{\partial C_i^t(\tau_i)}{\partial \tau_i} = 0$, we have

$$\tau_i^* = \sqrt{\frac{2\mu_e E_i^{\mathrm{total}}}{\mu_t}}. \tag{32}$$

Since the derivative of $C_i^t(\tau_i)$ is positive when $\tau_i > \tau_i^*$ and $C_i^t(\tau_i)$ is monotonic increasing, the optimal sample interval will be the lower bound if $\tau_i^* < \tau_{min}$. Hence, the optimal sampling interval is expressed as

$$\tau_i^* = \begin{cases} \sqrt{\frac{2\mu_e E_i^{\mathrm{total}}}{\mu_t}}, & \text{if } \sqrt{\frac{2\mu_e E_i^{\mathrm{total}}}{\mu_t}} > \tau_{min}, \\ \tau_{min}, & \text{otherwise.} \end{cases} \tag{33}$$

B. Sensing Time Optimization

In this part, the subproblem (28) is considered to determine the optimal number of the sensing time. Note that the value of $T_i^{1,\mathrm{prcs}}$ and P_i increases with s_i and the value of $C_i(s_i)$ increases with $T_i^{1,\mathrm{prcs}}$ and decreases with P_i. Besides, when the offloading policy is fixed, the costs of transmission and processing are determined. From the analysis above, the value of system overhead is rewritten with respect to the sensing time as

$$\mathcal{P}_2: \min_{S} \sum_{i=1}^{N} C_i^s(s_i) \tag{34}$$
$$= \sum_{i=1}^{N} \left(\mu_t \frac{T_i^{\mathrm{ses}}(s_i) + T_i^{\mathrm{ex}}}{P_i(s_i)} + \mu_e \frac{E_i^{\mathrm{ses}}(s_i) + E_i^{\mathrm{ex}}}{P_i(s_i)\tau_i} \right)$$
$$= \sum_{i=1}^{N} \left(\mu_t \frac{t_i^{\mathrm{unit}} s_i + T_i^{\mathrm{ex}}}{1-(1-p_i)^{s_i}} + \mu_e \frac{e_i d_i s_i + E_i^{\mathrm{ex}}}{\tau_i \left[1-(1-p_i)^{s_i}\right]} \right)$$
$$s.t.\ (26d). \tag{35}$$

where $T_i^{\mathrm{ex}} = \left(T_i^{\mathrm{trans}}(\mathbf{x}) + T_i^{\mathrm{edge}} \right) x_i + T_i^{\mathrm{local}}(1-x_i)$ and $E_i^{\mathrm{ex}} = E_i^{\mathrm{tran}}(\mathbf{x}) x_i + E_i^{\mathrm{local}}(1-x_i)$ are constant when the sampling interval and computation offloading policy remain unchanged.

To minimize the system overhead, the variation trend of the objective function needs to be considered. To change the non-convex feasible set into a convex set, we relax the discrete variable s_i into real value as $s_i \in [0, +\infty]$. It can be verified that the function of $C_i^s(s_i)$ is convex. Therefore, the value of $C_i^s(s_i)$ first decreases with s_i and increases with the increment of s_i and there is only one optimal solution for s_i. However, it is hard to achieve the optimal sensing time by directly solving $\frac{\partial C_i^s(s_i)}{\partial s_i} = 0$. Considering the sensing time is discrete and has an upper bound, an enumerating algorithm is proposed to find the optimal sensing time, which is shown in Algorithm 1. For each IoT device, the sensing time is initially set as $s_i = 1$. The number of sensing operations keeps increasing until the system overhead is no longer decreasing. Considering the computation complexity of Algorithm 1 is dependent on the sampling interval which is a constant and the number of IoT devices. Let $\overline{\tau}$ be the average value of the sampling interval. The optimal sensing time s_i^* can be derived with the complexity no more than $\mathcal{O}(\overline{\tau} N)$.

Algorithm 1 Enumerating for Sensing Time Optimization

Input: $\mathbf{x}$, T.
Output: Optimal sensing times S.

```
for each IoT device i ∈ N do
    Initialization: s_i = 1;
    Computing C_i(s_i)|_{s_i=1};
    while s_i ≤ ⌊τ_i / t_i^unit⌋ do
        if C_i(s_i + 1) < C_i(s_i) then
            s_i = s_i + 1;
        else
            break;
        end if
    end while
end for
```

C. Computation Offloading Optimization

In this part, subproblem $\mathcal{P}_3$ is solved to determine the optimal computation offloading policy with the aim of minimizing the system cost. From the conclusion of [39], the computation offloading decision-making problem can be transformed into the maximum cardinality bin packing problem [40], which is NP-hard. Therefore, finding a central solution to the subproblem $\mathcal{P}_3$ is NP-hard. In view of the complexity of the offloading computation optimization problem, game theory is introduced to provide a decentralized way to conduct the computation offloading decision-making.

Before solving the offloading problem, the data size constraint of the MEC server needs to be considered. Since the computation capacity of the MEC server is limited in practice, the number of data that can be processed at the same time is finite. When the data size of the MEC server exceeds the threshold, the MEC server will not be able to serve the IoT devices anymore. To meet the data constraint proposed in (26f), we design an MEC server availability request mechanism, which can be utilized to ensure a rational computation offloading decision without violating the data constraint of the MEC server.

Specifically, when an IoT device finishes the sensing operation and generates a status update, the IoT device first makes a preliminary offloading decision based on the requirement of its computing task and the network condition. Then, the IoT device will send a computation request with the data size of the status update to the MEC server, in order to acquire the equivalent computation capability of the MEC server. When the MEC server receives the request, the MEC server first summarizes all its computation tasks to determine whether its computation capacity upper bound is exceeded and the amount of spare computation capacity to be allocated. If $\sum_{i=1}^{N} x_i d_i \leq D_e$, the MEC server will permit the computation offloading request and send the computation capacity f_e to the IoT device. Otherwise, the data size of computing tasks transmitted to the MEC server is beyond the data threshold of the MEC server. To reasonably allocate the computing resource of the MEC server, the MEC server can list the IoT devices that are being served by the MEC server and eliminate the computing task with the most value of the data size continuously until the data threshold D_e is satisfied. For those IoT devices that are eliminated from the service list, the MEC server will send a message with the assigned value 0 of computation capacity to the deleted IoT devices. Consequently, the processing time for those IoT devices is infinite with 0 allocated computation capacity, and the IoT devices will choose local processing instead. Through the MEC server availability request mechanism, the MEC server unavailability is addressed and the subproblem $\mathcal{P}_3$ is transformed into the offloading decision-making problem. Due to the data size of messages is relatively small, the communication overhead caused by the availability request mechanism can be ignored.

Then, we formulate the computation offloading problem as a computation offloading game. Let $x_{-i} = \{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N\}$ be the computation offloading policy of the other IoT devices except for i. With the knowledge of the offloading strategies of other IoT devices, the IoT device i performs the offloading decision-making to minimize the system cost, i.e.

$$\min_{x_i \in \mathbf{x}} C_i(x_i, x_{-i}), \quad \forall i \in \mathcal{N}. \tag{36}$$

The offloading decision-making problem can be formulated as a strategic game $\Gamma = \{\mathcal{N}, \mathbf{x}, C_i\}$, where the IoT device set $\mathcal{N}$ is the set of players, $\mathbf{x}$ is the set of strategies taken by players, and the system cost $C_i(x_i, x_{-i})$ is the objective function to be minimized. Then, we define the Nash equilibrium of the game Γ as in the following definition.

Definition 1: A computation offloading strategy $\mathbf{x}^* = \{x_1^*, \ldots, x_N^*\}$ *is a Nash equilibrium if no IoT devices can further reduce the system overhead by unilaterally changing its own computation offloading strategy, i.e.,*

$$C_i(x_i^*, x_{-i}^*) \leq C_i(x_i, x_{-i}^*), \quad \forall x_i \in \{0, 1\}, \ \forall i \in \mathcal{N}. \tag{37}$$

For a multi-user computation offloading game, the Nash equilibrium guarantees that each IoT device at the Nash equilibrium achieves a mutually satisfactory policy and has no incentive to deviate from its original strategy. The property is because if any IoT device is about to change its offloading policy, it should obtain lower system cost by updating offloading policy, which is contradictory to the definition of Nash equilibrium. Then, we define the best response for each IoT device as follows.

Definition 2: For IoT device i, the strategy x_i^* is the best response based on the policies of other users x_{-i}, if the system cost satisfies that

$$C_i(x_i^*, x_{-i}^*) \leq C_i(x_i, x_{-i}^*), \quad \forall x_i \in \{0, 1\}. \tag{38}$$

To achieve the Nash equilibrium, all the IoT devices tend to take the best response strategy. Then we have following **Lemma 1**.

Lemma 1: An IoT device will achieve the lower system cost by offloading computing task to the MEC server for processing based on the offloading strategy $\mathbf{x}$*, if the received interference meets* $\sum_{m \in \mathcal{N}, m \neq i} x_m g_{m,s} p_m \leq L_i$*, where* L_i *is denoted as*

$$L_i = \frac{g_{i,s} p_i}{2^{\frac{B}{\mu_t d_i \tau_i + \mu_e p_i d_i} \left[\mu_t \tau_i \left(T_i^{local} - T_i^{edge} \right) + \mu_e E_i^{local} \right]} - 1} - \omega_0.$$

Proof: According to (13), (23) and (24), the lower system cost of transmitting computing tasks to the MEC server for processing is equivalent to

$$\mu_t \frac{T_i^{\text{local}}}{P_i} + \mu_e \frac{E_i^{\text{local}}}{P_i \tau_i} \geq \mu_t \frac{T_i^{\text{edge}} + T_i^{\text{tran}}(\mathbf{x})}{P_i} + \mu_e \frac{E_i^{\text{trans}}(\mathbf{x})}{P_i \tau_i}.$$

That is

$$r_i(\mathbf{x}) \geq \frac{\mu_t d_i \tau_i + \mu_e p_i d_i}{\mu_t \tau_i \left(T_i^{\text{local}} - T_i^{\text{edge}} \right) + \mu_e E_i^{\text{local}}}$$

Then, we can derive the threshold of the interference

$$\sum_{m \neq i} x_m g_{m,s} p_m \leq \frac{g_{i,s} p_i}{2^{\frac{B}{\mu_t d_i \tau_i + \mu_e p_i d_i} \left[\mu_t \tau_i \left(T_i^{\text{local}} - T_i^{\text{edge}} \right) + \mu_e E_i^{local} \right]} - 1} - \omega_0.$$

Accordingly, the best response of the IoT device i can be expressed as

$$x_i^* = \begin{cases} 1, & \text{if } \sum_{m \in \mathcal{N} \setminus \{i\}, x_m = 1} g_{m,s} p_m \leq L_i, \\ 0, & \text{otherwise.} \end{cases} \tag{39}$$

Based on Lemma 1, the computation offloading strategy of IoT device i is mainly dependent on its own received interference. To prove the existence of the Nash equilibrium in our proposed computation offloading game, we introduce the concept of the potential game [41].

Definition 3: A strategic game is called a potential game only if the variation of the utility function is proportional to the change of a certain function which is called potential function, i.e. there exists a potential function $\Phi(\mathbf{x})$ *satisfying that*

$$C_i(x_i, x_{-i}) < C_i(x_i', x_{-i}), \quad \textit{iff } \Phi(x_i, x_{-i}) < \Phi(x_i', x_{-i}) \tag{40}$$

for each IoT device $i \in \mathcal{N}$*, and any* $x_i, x_i' \in \mathbf{x}$.

Algorithm 2 Decentralized Computation Offloading Optimization Algorithm

Input: S, T.
Output: Optimal computation offloading strategy $\mathbf{x}^*$.

Initialize the computation offloading strategy that each IoT device chooses to process its task locally, i.e. $x_i = 0,\ \forall i \in \mathcal{N}$;
repeat for each iteration slot t
  Initialize the update set $\kappa = \emptyset$.
  for each IoT device $i \in \mathcal{N}$ **do**
    Select the best response x_i^* for i according to (39);
    if $x_i \neq x_i^*$ **then**
      Add IoT device i into the update set κ to compete for the updating opportunity;
    end if
  end for
  if $\kappa = \emptyset$ **then**
    break;
  end if
  for each IoT device i in the update set κ **do**
    Compute the improvement in the system cost by (41);
  end for
  The IoT device i with the most improvement in the system cost, Update the offloading strategy $x_i = x_i^*$;
  Broadcast the decision update to all the other devices;
  Other devices choose the original offloading strategy for next iteration slot $t+1$;
until $\kappa = \emptyset$ for several consecutive slots
return $\mathbf{x}^*$

Theorem 2: The computation offloading decision-making game is a potential game and always has at least one Nash equilibrium and possesses the finite improvement property.
Proof: Please see Appendix B.

According to Theorem 2, we can see that the computation offloading problem can achieve the Nash equilibrium after finite iterations. Next, we propose a decentralized computation offloading optimization algorithm in Algorithm 2 to achieve the mutually satisfactory offloading strategy for IoT devices.

To take the advantage of the finite improvement property of the potential game, we propose a decentralized computation offloading optimization algorithm to allow an IoT device to update its offloading strategy at one iteration. For each iteration, the update set is initialized as an empty set to record the IoT devices that have the incentive to update their offloading strategy. Based on the offloading strategies of other IoT devices x_{-i}, each IoT device computes its received interference by $\sum_{m\in\mathcal{N},m\neq i} x_m g_{m,s} p_m$. Then, the IoT devices select the best response strategy according to (39) and determine whether it needs to update its offloading strategy. If the best response is different from its current strategy, the device i will be added to the update set to compete for the opportunity to update the offloading strategy. After all the IoT devices decide their best responses, the devices in the update set will evaluate the improvement range of updating the offloading policy by

$$\Delta C_i = C_i(x_i^*, x_{-i}^*) - C_i(\overline{x_i^*}, x_{-i}^*), \tag{41}$$

where $\overline{x_i^*} = 1 - x_i^*$ is the original strategy of the device i. To improve the convergence speed of the iteration, the IoT device with the most improvement will win the competition and update its offloading strategy. The other devices will sustain their original offloading strategy and wait for the next iteration to contend for the updating opportunity. The offloading strategy will be continuously iterated until no device tends to update its offloading strategy for several consecutive iterations, and the optimal offloading policy $\mathbf{x}^*$ is obtained. Since the most operations in Algorithm 2 are basic mathematical calculations, the computational complexity of one iteration is mainly dependent on the sort of the device with the most improvement. Since each device needs to perform the sorting operation, therefore the complexity of one iteration is $\mathcal{O}(N \log N)$. Assuming that I iterations are required to achieve the Nash equilibrium, the complexity of Algorithm 2 is $\mathcal{O}(IN \log N)$.

D. Algorithm Summary

In this subsection, we summarize the multi-variable iteration system cost optimization algorithm (MISCO) for joint sensing and processing optimization to minimize the system overhead. To solve the overall optimization problem (26a), we execute the iterations of the sensing, transmission and computation offloading optimization. First, we solve the optimal sampling period for each IoT device. Afterwards, based on the upper bound of the sampling interval, we utilize the enumeration method to determine the optimal sensing time. Based on the result of the sampling and sensing optimization, a game-theoretic optimization algorithm is proposed to solve the optimal computation offloading strategy. Iterations of the sampling interval, sensing time and computation offloading optimization terminate when the disparity of the overall system cost $\hat{C} = \sum_{i=1}^{N} C_i$ between two consecutive iterations is below the threshold ϵ. The details of the proposed algorithm are summarized in Algorithm 3. Based on the analysis above, the sampling interval set, the sensing time set and the computation offloading strategy are updated during the iteration process and the overall system cost keeps decreasing in each iteration. Considering the system cost has a lower bound and can only decrease finitely, the proposed multi-variable iterative optimization algorithm is convergent. Assuming K iterations are requisite to meet the disparity threshold, the complexity of Algorithm 3 can be expressed as $\mathcal{O}(KN + K\bar{\tau}N + KIN \log N)$.

VII. Simulation Results

In this section, we evaluate the performance of our proposed system overhead minimization algorithm by numerical results. We assume the coverage of the MEC server is a 50 m $\times$ 50 m area and N IoT devices are randomly distributed in the coverage area to execute the sensing tasks and generate the status updates. The channel gain of each IoT device is calculated as $g_{i,s} = v^{-o}$, where v is the distance between the IoT device and

Algorithm 3 Multi-Variable Iterative System Cost Optimization Algorithm

Input: Status update set U, distance set D, computation capacity f_i and f_e, ω_0, channel gain g, transmission power p, sensing time unit t_i^{unit}.

Output: Sampling interval T^*, sensing time set S^*, optimal computation offloading strategy $\mathbf{x}^*$.

1: Set $r = 0$ as the iteration slot. Initialize the sampling interval set $\mathbf{T}^0$, the sensing time set S^0 and the computation offloading strategy profile $\mathbf{x}^0$ randomly;
2: **repeat** for each iteration slot r
3: Given the fixed S^r and $\mathbf{x}^r$, solve the sampling interval $\mathbf{T}^{r+1}$ according to (33);
4: Given the $\mathbf{T}^{r+1}$ and $\mathbf{x}^r$, solve the sensing time S^{r+1} using Algorithm 1;
5: Given the $\mathbf{T}^{r+1}$ and S^{r+1}, solve the offloading strategy $\mathbf{x}^{r+1}$ using Algorithm 2;
6: Compute the overall system cost $\hat{C}^{r+1}$ based on $\mathbf{T}^{r+1}$, S^{r+1} and $\mathbf{x}^{r+1}$;
7: $r = r + 1$;
8: **until** $|\hat{C}^r - \hat{C}^{r-1}| < \epsilon$
9: **return** $\mathbf{x}^*$

TABLE I
PARAMETERS USED FOR SIMULATION

Parameters	Values
t_i^{unit}	0.2s
ϵ	0.08
e_i	10^{-9} Joules/bit
d_i	500 KB
c_i	1000 Megacycles
B	100Mhz
p_i	100mW
ω_0	-100 dBm
f_i	$[0.8, 1.0]$ Ghz
f^e	20 Ghz
δ	$10^{-11}(f_i)^2$

the MEC server and o is the path loss coefficient set as 4. The other parameters used in the simulation are given in Table I.

Given the lack of a joint optimization scheme for task sensing and processing, We evaluate the performance of our proposed optimization scheme by comparing the impact of each variable individually. To ensure a fair and accurate comparison, all variables are held constant, with the exception of the variable under comparison. Accordingly, the specific effect of each variable on the overall performance can be isolated and analyzed. Specifically, the influence of sampling frequency is examined through the comparison with GSA, the influence of number of the sensing operations is displayed by comparing with ISA and the comparison with BRCO, SGDCO, and AECO shows the optimization in the computation offloading strategy. Then, the overall performance of our proposed optimization method can be attained. Therefore, we introduce five comparative algorithms as benchmarks:

- Greedy Sensing Algorithm (GSA): The sampling interval is decided according to (33). The IoT devices will execute the least sensing operations to meet the sensing successful probability to achieve the least sensing latency. Then, the offloading decision-making is made by Algorithm 2.
- Instant Sampling Algorithm (ISA): Each time when the previous status update finishes processing, the IoT device will conduct another sampling process to generate a new status update, which is similar to the zero-wait policy in [42]. The sensing time decision is made by Algorithm 1 and the computation offloading strategy is made by Algorithm 2.
- Best Response Computation Offloading (BRCO) [34]: The sampling interval is decided by (33), and the sensing time is obtained from Algorithm 1. Each device chooses the best-response strategy based on the computing cost of the two processing ways and the offloading probabilities of other devices in the previous stage.
- Stochastic Gradient Descent Computation offloading (SGDCO): SGDCO utilizes the stochastic gradient descent to solve the computation offloading problem while the optimization of sensing time and sensing interval is the same as MISCO [43].
- All Offloading to Edge Computing (AECO): The sampling interval and sensing time are determined in the same way as our proposed optimization algorithm. However, all the computing tasks are offloaded to the MEC server for processing.

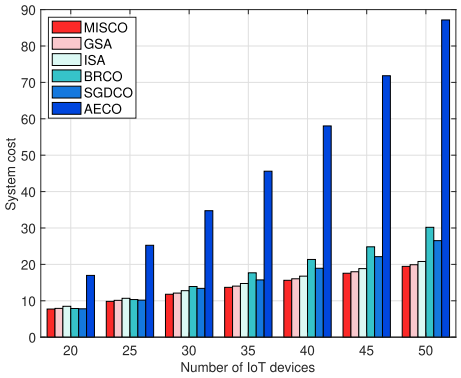

Fig. 4. System cost vs. number of IoT devices.

We first evaluate the system cost of our proposed method. Fig. 4 shows the system cost of different methods with the different numbers of IoT devices. Compared to the other schemes, our proposed MISCO achieves the lowest system cost no matter what the number of IoT devices is. When the number of IoT devices is relatively small, there is not much difference between AECO and the other five comparative methods due to the small interference caused by IoT devices. Compared to the other three computation offloading optimization methods BRCO, SGDCD and AECO, our proposed method has a better performance in system cost as the number of IoT devices increases. With the larger number of IoT devices, the bandwidth for IoT devices to execute the task transmission is insufficient and the transmission cost improves greatly. Therefore, our computation offloading method indicates a more rational offloading decision-making process.

Fig. 5 shows the system cost with the different numbers of the CPU cycles required to process the status update.

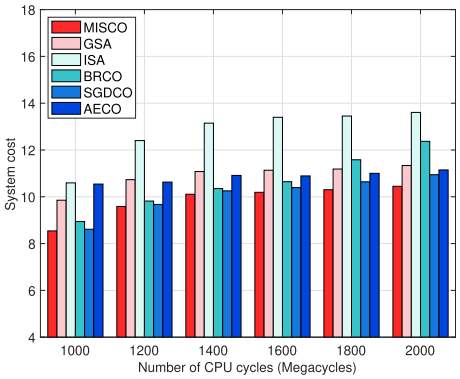

Fig. 5. System cost vs. number of CPU processing cycles.

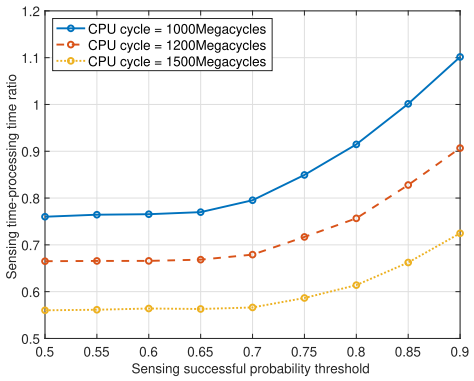

Fig. 6. Sensing time-processing time ratio vs. successful sensing probability thresholds.

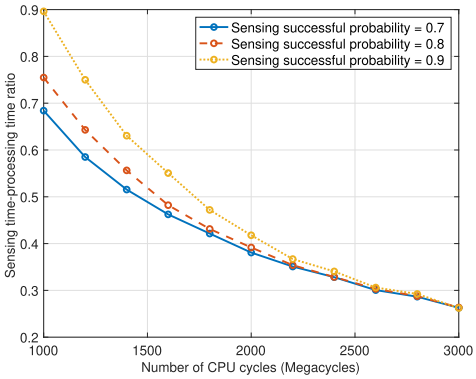

Fig. 7. Sensing time-processing time ratio vs. number of CPU processing cycles.

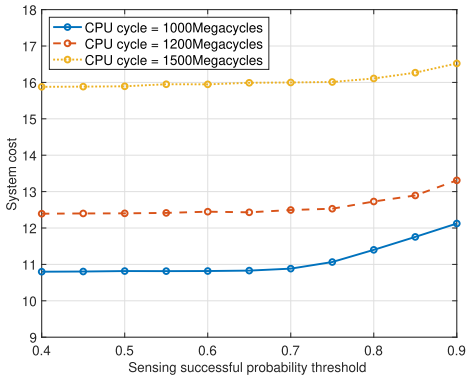

Fig. 8. System cost vs. successful sensing probability thresholds.

The numerical result shows the system cost of MISCO is lower than other benchmarks with more computation load of the status update. The system cost of ISA rises significantly because the longer process time of the status update makes the status sampling out of date, while the increment of system cost of sensing optimization is relatively stable as the number of CPU cycles increases. With the improvement of the CPU cycle, the performance of MISCO is close to the one of AECO. Due to the heavy computation load, all IoT devices choose to offload their tasks.

Fig. 6 depicts the sensing time-processing time ratio with the different successful sensing probability under various computation loads. Note that the processing time here consists of the transmission time and the task processing time. When the sensing successful probability threshold is relatively small, the sensing time-processing time ratio remains at a stable level. Considering the threshold is easy to meet, the sensing time is determined by the sensing optimization method. As the requirement of sensing successful probability threshold increases, more sensing operations need to be performed to satisfy the successful probability threshold, which causes a longer sensing time for IoT devices. Specifically, when the CPU cycle is 1000 Megacycles and the sensing successful probability threshold is more than 0.65, the sensing successful probability threshold has a great influence on the sensing-processing ratio. That is, the processing time dominates the sensing time-processing time ratio when the successful probability requirement is not strict and then the sensing time dominates with the high successful probability requirement. As the CPU cycle increases, the sensing time dominates at a higher level of successful probability due to the more processing time for status updates. When the CPU cycle is 1500 Megacycles, the sensing-processing ratio begin to increase when the sensing successful probability threshold is more than 0.7.

The simulation result of the sensing time-processing time ratio with the different CPU cycles is shown in Fig. 7. When the computation load is small, the sensing time dominates the sensing-processing ratio. With the higher sensing successful probability threshold, the sensing time accounts for a higher proportion of the total time cost. As the number of CPU cycles increases, the sensing time as a percentage of total time decreases sharply. In the case of large number of CPU cycles, the sensing successful probability has little effect on the ratio of sensing time to processing time. The sensing-processing ratio remains at the same level for different successful probability thresholds, and it can be seen that processing time accounts for the major part of the total time at a high computation load.

Fig. 8 shows the impact of different sensing successful probability thresholds on the system cost. When the threshold is at a low level, the system cost will not be limited by the threshold and can be optimized to get the minimal system overhead directly by our proposed optimization algorithm. Therefore, the system cost will be maintained at a stable level. After the threshold value exceeds 0.7, the optimization method

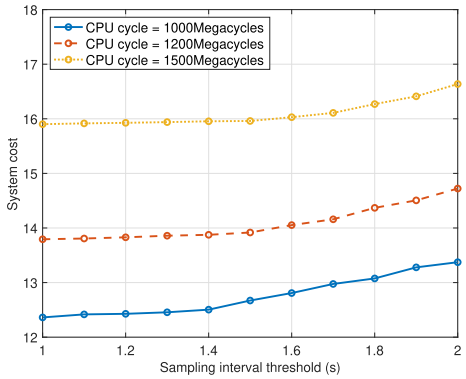

Fig. 9. System cost vs. sampling interval thresholds.

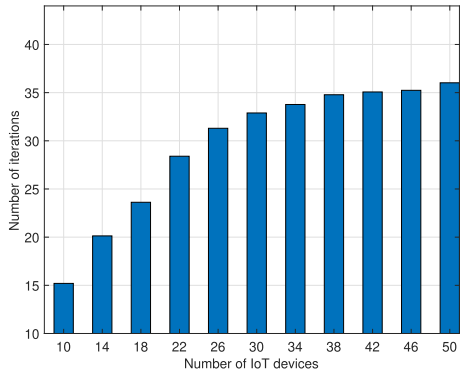

Fig. 10. Number of iterations vs. Number of IoT devices.

will be limited to achieve the optimal system overhead in order to meet the sensing successful rate requirement. When the CPU cycles for the task execution are small, the more obvious is the influence of the sensing successful rate threshold and the system cost rises more obviously. Therefore, the optimal threshold should be set to about 0.7 to ensure the sensing quality as well as the value of the system overhead.

In Fig. 9, the numerical result shows the system cost of our proposed MISCO with different sampling intervals. With the high sampling frequency limit, the system overheads for different task computations loads are sustained at a stable value. When the sampling interval is greater than 1.4 s, the system overhead starts to increase gradually. This means that the sampling interval limit at this point is greater than the optimal sampling interval solved by our proposed algorithm, and the excessively long sampling interval leads to an increase in the AoI, resulting in an increase in system overhead. To reduce the energy consumption of generating status updates and the AoI of status updates, the sampling interval threshold should be set lower than a certain level.

Fig. 10 depicts the number of iterations of our proposed method with the different numbers of IoT devices. Here the number of iterations contains the iteration number of Algorithm 2 to achieve the Nash equilibrium and the iteration number of Algorithm 3 for converging. The iteration number increases with the increased number of IoT devices, which illustrates the convergence and scalability of our proposed MISCO. When the number of IoT devices is getting larger, the network resources are insufficient for the IoT devices. Therefore, all the IoT devices tend to process their computing tasks locally which makes the number of iterations reach an upper bound due to all devices stop iterating after choosing local execution.

VIII. Conclusion

In this paper, we first formulate the joint sensing and processing optimization problem to optimize the system performance including the information freshness of the status updates and the energy consumption of IoT devices. The optimization problem is decomposed into three subproblems to optimize the sampling, sensing and computation offloading respectively. The sampling and sensing optimization problems are solved by extremum principles and game-theoretic method is utilized to perform the computation offloading decision-making. Afterwards, the multi-variable iterative optimization algorithm is proposed to minimize the system cost jointly. Numerical results depict that the system cost achieved by our proposed method is lower than other comparative methods and the dominance of sensing and processing under different scenarios. Besides, the impact of the sensing probability and sampling interval thresholds are analyzed in the simulation.

Appendix A
Proof of Theorem 1

The probability that IoT device i requires j sensing operations to generate a valid status update is $P_i(s_i)(1-P_i(s_i))^{j-1}$. Thus the expectation of execution time can be calculated as

$$\begin{aligned}\mathbb{E}[T_i^{\text{prcs}}] = &\lim_{j\to\infty} P_i(s_i)\, T_i^{1,\text{prcs}}(s_i,\mathbf{x}) \\ &+ P_i(s_i)(1-P_i(s_i))T_i^{2,\text{prcs}}(s_i,\mathbf{x}) \\ &+ P_i(s_i)(1-P_i(s_i))^2 T_i^{3,\text{prcs}}(s_i,\mathbf{x}) \\ &+ \ldots + P_i(s_i)(1-P_i(s_i))^{j-1} T_i^{j,\text{prcs}}(s_i,\mathbf{x}) \\ = &\lim_{j\to\infty} P_i(s_i)\, T_i^{1,\text{prcs}}(s_i,\mathbf{x}) \sum_{n=1}^{j} n(1-P_i(s_i))^{n-1}.\end{aligned} \tag{42}$$

Given the value of $P_i(s_i)$ is in the range of (0,1), let $1-P_i(s_i)$ be a single variable ρ and $\sum_{n=1}^{\infty} n(1-P_i(s_i))^{n-1}$ can be further calculated as

$$\begin{aligned}\sum_{n=1}^{\infty} n(1-P_i(s_i))^{n-1} &= \sum_{n=1}^{\infty} [(1-P_i(s_i))^n]' \\ &= \sum_{n=1}^{\infty} (\rho^n)' = \left(\frac{\rho}{1-\rho}\right)' \\ &= \frac{1}{(1-\rho)^2} = \frac{1}{P_i(s_i)^2}.\end{aligned} \tag{43}$$

By substituting (43) into the expectation of execution time, the average number of processing time for a successful status update is calculated as $\mathbb{E}[T_i^{\text{prcs}}] = \frac{T_i^{1,\text{prcs}}(s_i,\mathbf{x})}{P_i(s_i)}$.

APPENDIX B
PROOF OF THEOREM 2

To prove the computation offloading decision-making game is a potential game, we define the potential function as

$$\Phi(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{m\neq i} g_{i,s}p_i x_i g_{m,s}p_m x_m + \sum_{i=1}^{N} g_{i,s}p_i L_i(1-x_i). \tag{44}$$

We consider an IoT device that chooses to update its offloading policy with a lower system overhead, i.e. $C_i(x_i, x_{-n}) < C_i(x_i', x_{-n})$. From the definition of the potential game, the decrease of the system cost will lead to a decrease of the potential function. If the original offloading policy is to process the task locally, i.e. $x_i' = 0,\ x_i = 1$, we derive $C_i(1, x_{-n}) < C_i(0, x_{-n})$, and $\sum_{m\in\mathcal{N}\setminus\{i\}, x_m=1} g_{m,s}p_m \leq L_i$ is meet. Then, we compute the change of the potential function by updating the offloading policy:

$$\begin{aligned}
&\Phi(1, x_{-i}) - \Phi(0, x_{-i}) \\
&= \frac{1}{2}\sum_{j\neq i}\sum_{m\neq i, m\neq j} g_{j,s}p_j x_j g_{m,s}p_m x_m + \frac{1}{2}g_{i,s}p_i\sum_{j\neq i} g_{j,s}p_j x_j \\
&\quad + \frac{1}{2}g_{i,s}p_i\sum_{m\neq i} g_{m,s}p_m x_m + \sum_{j\neq i} g_{j,s}p_j L_j(1-x_j) \\
&\quad - \frac{1}{2}\sum_{j\neq i}\sum_{m\neq i, m\neq j} g_{j,s}p_j x_j g_{m,s}p_m x_m \\
&\quad - \sum_{j\neq i} g_{j,s}p_j L_j(1-x_j) - g_{i,s}p_i L_i \\
&= g_{i,s}p_i\sum_{m\neq i} g_{m,s}p_m x_m - g_{i,s}p_i L_i < 0.
\end{aligned} \tag{45}$$

For $x_i' = 1, x_i = 0$, the result is similar to the argument above. According to the definition of the potential game, we conclude that the computation offloading decision-making problem is a potential game.

REFERENCES

[1] C. Xu, Y. Xie, X. Wang, H. H. Yang, D. Niyato, and T. Q. S. Quek, "Optimal status update for caching enabled IoT networks: A dueling deep R-network approach," *IEEE Trans. Wireless Commun.*, vol. 20, no. 12, pp. 8438–8454, Dec. 2021.

[2] F. Peng, Z. Jiang, S. Zhou, Z. Niu, and S. Zhang, "Sensing and communication co-design for status update in multiaccess wireless networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 3, pp. 1779–1792, Aug. 2021.

[3] T. Zhang, Y. Xu, J. Loo, D. Yang, and L. Xiao, "Joint computation and communication design for UAV-assisted mobile edge computing in IoT," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 5505–5516, Oct. 2019.

[4] I. A. Elgendy, W.-Z. Zhang, Y. Zeng, H. He, Y.-C. Tian, and Y. Yang, "Efficient and secure multi-user multi-task computation offloading for mobile-edge computing in mobile IoT networks," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 4, pp. 2410–2422, Dec. 2020.

[5] 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, 4th Quart., 2017.

[6] Z. Ning et al., "Dynamic computation offloading and server deployment for UAV-enabled multi-access edge computing," *IEEE Trans. Mobile Comput.*, early access, Nov. 23, 2021, doi: 10.1109/TMC.2021.3129785.

[7] F. Wu, C. Qiu, T. Wu, and M. R. Yuce, "Edge-based hybrid system implementation for long-range safety and healthcare IoT applications," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9970–9980, Jun. 2021.

[8] O. Elijah, T. A. Rahman, I. Orikumhi, C. Y. Leow, and M. N. Hindia, "An overview of Internet of Things (IoT) and data analytics in agriculture: benefits and challenges," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3758–3773, Oct. 2018.

[9] C. Lee and S.-L. Kim, "Most efficient sensor network protocol for a permanent natural disaster monitoring system," *IEEE Internet Things J.*, vol. 8, no. 15, pp. 11776–11792, Aug. 2021.

[10] S. Kaul, M. Gruteser, V. Rai, and J. Kenney, "Minimizing age of information in vehicular networks," in *Proc. 8th Annu. IEEE Conf. Sensor, Mesh Ad Hoc Commun. Netw. (SECON)*, Salt Lake City, UT, USA, Jun. 2011, pp. 350–358.

[11] Y. Sang, B. Li, and B. Ji, "The power of waiting for more than one response in minimizing the age-of-information," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Singapore, Dec. 2017, pp. 1–6.

[12] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[13] A. Makhoul, H. Harb, and D. Laiymani, "Residual energy-based adaptive data collection approach for periodic sensor networks," *Ad Hoc Netw.*, vol. 35, pp. 149–160, Dec. 2015.

[14] L. Hu, Z. Chen, D. Deng, M. Wang, Y. Jia, and T. Q. S. Quek, "Joint optimization of freshness and fidelity for status updates in a periodical decision system," *IEEE Trans. Veh. Technol.*, vol. 72, no. 3, pp. 3569–3583, Mar. 2023.

[15] Y. H. Bae and J. W. Baek, "Age of information and throughput in random access-based IoT systems with periodic updating," *IEEE Wireless Commun. Lett.*, vol. 11, no. 4, pp. 821–825, Apr. 2022.

[16] S. Zhang, H. Zhang, Z. Han, H. V. Poor, and L. Song, "Age of information in a cellular Internet of UAVs: Sensing and communication trade-off design," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6578–6592, Oct. 2020.

[17] R. Li, Q. Ma, J. Gong, Z. Zhou, and X. Chen, "Age of processing: Age-driven status sampling and processing offloading for edge-computing-enabled real-time IoT applications," *IEEE Internet Things J.*, vol. 8, no. 19, pp. 14471–14484, Oct. 2021.

[18] C. Kam, S. Kompella, G. Nguyen, J. Wieselthiery, and A. Ephremides, "Towards an effective age of information: Remote estimation of a Markov source," in *Proc. IEEE INFOCOM 1st Workshop Age-of-Inf.*, Honolulu, HI, USA, 2018, pp. 1–9.

[19] O. Ayan, M. Vilgelm, M. Klügel, S. Hirche, and W. Kellerer, "Age-of-information vs. value-of-information scheduling for cellular networked control systems," in *Proc. 10th ACM/IEEE Int. Conf. Cyber-Phys. Syst.*, Apr. 2019, pp. 109–117.

[20] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.

[21] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.

[22] R. D. Yates and S. K. Kaul, "The age of information: Real-time status updating by multiple sources," *IEEE Trans. Inf. Theory*, vol. 65, no. 3, pp. 1807–1827, Mar. 2019.

[23] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1183–1210, May 2021.

[24] I. Kadota, A. Sinha, and E. Modiano, "Optimizing age of information in wireless networks with throughput constraints," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Honolulu, HI, USA, Apr. 2018, pp. 1844–1852.

[25] S. Feng and J. Yang, "Age of information minimization for an energy harvesting source with updating erasures: Without and with feedback," *IEEE Trans. Commun.*, vol. 69, no. 8, pp. 5091–5105, Aug. 2021.

[26] B. Zhou and W. Saad, "Joint status sampling and updating for minimizing age of information in the Internet of Things," *IEEE Trans. Commun.*, vol. 67, no. 11, pp. 7468–7482, Nov. 2019.

[27] X. Chen et al., "Age of information aware radio resource management in vehicular networks: A proactive deep reinforcement learning perspective," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2268–2281, Apr. 2020.

[28] J. Zhao et al., "Secure resource allocation for UAV assisted joint sensing and comunication networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Austin, TX, USA, Apr. 2022, pp. 1856–1861.

[29] K. Liu and J. Zheng, "UAV trajectory optimization for time-constrained data collection in UAV-enabled environmental monitoring systems," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 24300–24314, Dec. 2022.

[30] J. Hu, H. Zhang, L. Song, R. Schober, and H. V. Poor, "Cooperative Internet of UAVs: Distributed trajectory design by multi-agent deep reinforcement learning," *IEEE Trans. Commun.*, vol. 68, no. 11, pp. 6807–6821, Nov. 2020.

[31] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

[32] Y. Yang, C. Long, J. Wu, S. Peng, and B. Li, "D2D-enabled mobile-edge computation offloading for multiuser IoT network," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12490–12504, Aug. 2021.

[33] Y. Ding, K. Li, C. Liu, and K. Li, "A potential game theoretic approach to computation offloading strategy optimization in end-edge-cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 6, pp. 1503–1519, Jun. 2022.

[34] Y. Wang et al., "A game-based computation offloading method in vehicular multiaccess edge computing networks," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4987–4996, Jun. 2020.

[35] Y. Mao, J. Zhang, Z. Chen, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[36] L. Corneo, C. Rohner, and P. Gunningberg, "Age of information-aware scheduling for timely and scalable Internet of Things applications," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2019, pp. 2476–2484.

[37] V. V. Shakhov and I. Koo, "Experiment design for parameter estimation in probabilistic sensing models," *IEEE Sensors J.*, vol. 17, no. 24, pp. 8431–8437, Dec. 2017.

[38] M. Sipser, *Introduction to the Theory of Computation*, 3rd ed. Boston, MA, USA: Cengage Learning, 2012, pp. 225–277.

[39] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2015.

[40] KH. Loh, B. Golden, and E. Wasil, "Solving the maximum cardinality bin packing problem with a weight annealing-based algorithm," in *Operations Research and Cyber-Infrastructure*. New York, NY, USA: Springer, 2009.

[41] D. Monderer and L. S. Shapley, "Potential games," *Games Econ. Behavior*, vol. 14, no. 1, pp. 124–143, May 1996.

[42] B. Barakat, S. Keates, I. Wassell, and K. Arshad, "Is the zero-wait policy always optimum for information freshness (peak age) or throughput?" *IEEE Commun. Lett.*, vol. 23, no. 6, pp. 987–990, Jun. 2019.

[43] Z. Liao, Y. Ma, J. Huang, J. Wang, and J. Wang, "HOTSPOT: A UAV-assisted dynamic mobility-aware offloading for mobile-edge computing in 3-D space," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10940–10952, Jul. 2021.

Yi Chen received the B.S. degree from the School of Computer and Software, Nanjing University of Information Science and Technology, in 2021. He is currently pursuing the M.S. degree with the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include mobile computing, edge computing, the IoT, cloud computing, and big data.

Zheng Chang (Senior Member, IEEE) received the Ph.D. degree from the University of Jyväskylä, Jyvä skylä, Finland, in 2013. He has published over 140 papers in journals and conferences, and received best paper awards from IEEE TCGCC and APCC in 2017. He has been awarded as the 2018 IEEE Communications Society Best Young Researcher for Europe, Middle East, and Africa Region, and the 2021 IEEE Communications Society MMTC Outstanding Young Researcher. He serves as an Editor for IEEE WIRELESS COMMUNICATIONS LETTERS, *China Communications*, and *International Journal of Distributed Sensor Networks*, and a Guest Editor for *IEEE Network*, IEEE WIRELESS COMMUNICATIONS, *IEEE Communications Magazine*, IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, *Physical Communications*, *EURASIP Journal on Wireless Communications and Networking*, and *Wireless Communications and Mobile Computing*. He was the Exemplary Reviewer of IEEE WIRELESS COMMUNICATION LETTERS in 2018. He has participated in organizing workshop and special session in Globecom' 19, WCNC'18-22, SPAWC'19, and ISWCS'18. He also serves as the Symposium Co-Chair for IEEE ICC'20 and Globecom'23, the Publicity Co-Chair for IEEE Infocom'22, the Workshop Co-Chair for ICCC'22, the TPC Co-Chair for IEEE iThing'22, and a TPC Member for many IEEE major conferences, such as INFOCOM, ICC, and Globecom. His research interests include the IoT, cloud/edge computing, security and privacy, vehicular networks, and green communications.

Geyong Min (Senior Member, IEEE) received the B.Sc. degree in computer science from the Huazhong University of Science and Technology, China, in 1995, and the Ph.D. degree in computing science from the University of Glasgow, U.K., in 2003. He is currently a Professor in high performance computing and networking with the Department of Computer Science, University of Exeter, U.K. His research interests include computer networks, wireless communications, parallel and distributed computing, ubiquitous computing, multimedia systems, modeling, and performance engineering.

Shiwen Mao (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from Polytechnic University, Brooklyn, NY, USA. He is currently a Professor and an Earle C. Williams Eminent Scholar and the Director of the Wireless Engineering Research and Education Center, Auburn University. His research interests include wireless networks and multimedia communications. He is also a Distinguished Lecturer of the IEEE Communications Society and IEEE Council of RFID. He was a co-recipient of the 2021 Best Paper Award of Elsevier/KeAi Digital Communications and Networks, the 2021 IEEE INTERNET OF THINGS JOURNAL Best Paper Award, the 2021 IEEE Communications Society Outstanding Paper Award, the IEEE Vehicular Technology Society 2020 Jack Neubauer Memorial Award, the 2004 IEEE Communications Society Leonard G. Abraham Prize in the Field of Communications Systems, and several best conference paper/demo awards. He is the Editor-in-Chief of IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING.

Timo Hämäläinen (Senior Member, IEEE) has over 25 year's experience of the computer networks. He has more than 250 internationally peer reviewed publications and he has supervised more than 40 Ph.D. theses. His research interests include performance evaluation and management of telecommunication networks, and in particular resource allocation, quality of service, anomaly detection, and network security. He is currently leading a research group in the area of network resource management and anomaly detection with the IT Faculty, University of Jyväskylä. He is a board member of this area's journals and IEEE conferences.