Freshness-aware Information Update and Computation Offloading in Mobile Edge Computing

Xiao Ma, *Member, IEEE,* Ao Zhou, *Member, IEEE* Qibo Sun, *Member, IEEE* and Shangguang Wang, *Senior Member, IEEE,*

***Abstract*—Mobile edge computing is a promising computing paradigm with the advantages of reduced delay and relieved outsourcing traffic to the core network. In mobile edge computing, reducing the computation offloading cost of mobile users and maintaining fresh information at edge nodes are two critical while conflicted objectives, as both consume the limited wireless bandwidth of edge nodes. Although extensive efforts have been devoted to optimizing computation offloading decisions and some works have investigated freshness-aware channel allocation issues recently, no prior works have considered the above conflict. This paper is the first work to jointly optimize the channel allocation and computation offloading decisions, aiming at reducing the computation offloading cost within freshness requirements of sensors. We analyze the recursiveness of AoI in analogy to the evolvement of a queue and formulate the problem as a nonlinear integer dynamic optimization problem. To overcome the challenges of AoI-computation cost tradeoff, AoI time dependency and high complexity caused by the heterogeneity of users, we propose an algorithm to solve the problem with reduced computation complexity. Specifically, we first transform the original problem into a static optimization problem in each time slot (which is NP-hard) based on Lyapunov optimization techniques. To reduce the computation complexity, we exploit the finite improvement property of potential games and further enforce centralized control to reduce the number of improvement iterations. Simulations have been conducted and the results demonstrate that the proposed algorithm shows good effectiveness and scalability.**

***Index Terms*—information update, channel allocation, computation offloading, edge computing**

I. Introduction

WITH the widespread deployment of IoT sensors and the proliferation of mobile devices, tremendous mobile data are being produced and a large amount of delay-sensitive and computation-intensive mobile tasks need to be processed. Mobile edge computing is considered as a promising computing paradigm by provisioning storage and computation resources (forming edge nodes) within the wireless access network [1], [2], [3]. Through caching mobile data of sensors and processing mobile tasks of mobile devices at edge nodes, mobile edge computing can on one hand relieve the traffic burden of the core network, and on the other hand, can effectively reduce the processing cost (in terms of delay and energy consumption) of mobile tasks. Due to the above advantages, mobile edge computing has attracted huge attention of researchers since its emergence.

Xiao Ma, Ao Zhou, Qibo Sun, and Shangguang Wang are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China, 100876. E-mail: maxiao18@bupt.edu.cn, aozhou@bupt.edu.cn, qbsun@bupt.edu.cn, sgwang@bupt.edu.cn. Ao Zhou is the corresponding author.

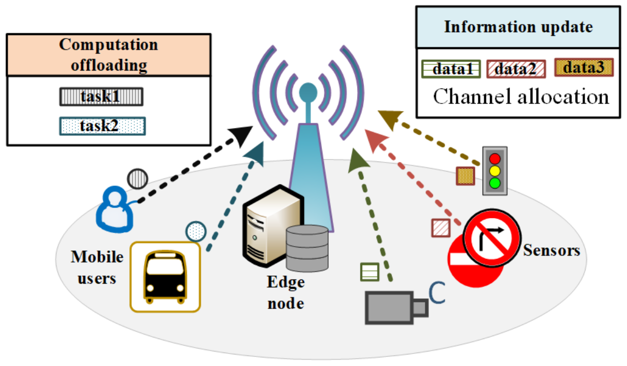


Fig. 1. Computation offloading and channel allocation in mobile edge computing.

As mobile edge computing can provision sufficient resources in proximity to mobile devices, offloading mobile tasks to edge nodes is potential to reduce the processing delay and energy consumption of mobile users, which is inherently advantageous for the fast-growing delay-sensitive and computation-intensive mobile tasks, such as AR/VR and object recognition. Thus, extensive research works have investigated computation offloading issues, aiming at reducing computation offloading cost in terms of delay and energy consumption [18]-[29]. In addition to the reduced computation offloading cost, some typical delay-sensitive and freshness-aware applications of mobile edge computing, such as real-world map navigation and obstacle avoidance for auto driving, also require fresh information maintained at edge nodes when processing the computation tasks. Therefore, maintaining fresh information at edge nodes is another vital objective for edge administrators to provision services for these types of applications. Age of information (AoI) has been proposed to depict the freshness of information and is defined as the time elapsed since its generation [4], [5]. Minimizing AoI at edge nodes requires frequent data update between sensors and edge nodes, occupying the limited wireless bandwidth of edge nodes. It is intuitive that there exists a conflict between maintaining fresh information at edge nodes and reducing the computation offloading cost of mobile users, as both consume the limited uplink wireless resources of edge nodes.

Although extensive works have been devoted to computation offloading and there have been some works investigating freshness-aware channel allocation recently, there are no prior works jointly optimizing the computation offloading and channel allocation decisions. Solving this problem is challenging. First, there exists a tradeoff between keeping information fresh and reducing computation offloading cost. To keep the information at the edge node fresh, mobile data of sensors should be updated frequently and more resources should be reserved for the information update channel. However, as the total bandwidth of the edge node is limited, more resources reserved for the information update channel means less resources left for computation offloading, leading to high computation offloading cost. Second, the future AoI is highly dependent on the state and channel allocation decision of the current time slot. Maintaining the long-term average AoI low requires to consider this time dependency when making the channel allocation decision for each time slot. Third, mobile users are heterogeneous since they have differentiated mobile computation capacities and diverse transmission power. To efficiently reduce the overall computation offloading cost of mobile users, the computation offloading decisions need to be well coordinated, which can cause high computation complexity.

In this paper, we investigate the freshness-aware information update and computation offloading problem in mobile edge computing. To our best knowledge, this is the first work to jointly optimize computation offloading and channel allocation decisions considering the tradeoff. We formulate this problem as a long-term dynamic optimization problem, aiming at minimizing the computation offloading cost of mobile users while ensuring the constraints of sensors. To analyze the constraints of sensors in depth, we combine the recursiveness of AoI with the evolvement of a queue and transform the throughput constraint by introducing virtual queues. Based on Lyapunov optimization techniques, the long-term information update and computation offloading strategy can be optimized by solving a static optimization problem in each time slot [6]. As the static optimization problem is proved to be NP-hard [7], we seek to reduce the complexity by exploiting the finite improvement property of potential games [8] and enforcing central control at the cost of information collection overhead. The contributions of this paper are summarized as follows.

- We are the first to jointly optimize the computation offloading and channel allocation decisions by considering the tradeoff between keeping fresh information at the edge node and reducing the computation offloading cost of mobile users. We analyze the recursive property of AoI in analogy to the evolvement of a queue and the transform the throughput constraints by introducing virtual queues.
- We formulate the problem as a nonlinear integer dynamic optimization problem and transform it into a static optimization problem in each time slot using Lyapunov optimization techniques. To solve the static optimization problem with reduced complexity, we exploit the finite improvement property of potential games and further enforce centralized control to reduce the number of improvement iterations.
- Extensive simulation are conducted and the results validate the effectiveness and scalability of the proposed algorithm.

The rest of the paper is organized as follows. Sec. II reviews the related work. In Sec. III, we present the system model and in Sec. IV, the problem formulation and algorithm design are provided. Sec. V illustrates the simulation results and Sec. VI concludes the paper.

II. Related Work

AoI has been proposed to depict the freshness of information, which has been considered as a critical performance metrics in research areas such as internet of things [9], [10], wireless network [11], [12], [13], [14], [15], [16], edge caching [4], [5], [17], and vehicular network [18]. Corneo *et al* in [10] have proposed an AoI-aware scheduling policy for sensor data updates in cloud caches, which performs well even with significant delay variations. Qian *et al* in [11] have studied AoI in muti-channel wireless systems. A policy-dependent lower bound has been presented and scheduling policies have been designed to provision good AoI performance. Champati *et al* in [12] have sought to derive the minimum AoI in a queue with single source and single server given the distribution of service time. Lou *et al* in [13] have explored the relationship between AoI and throughput of multi-hop wireless networks. Zhang *et al* in [17] have presented a threshold-based cache updating policy and have derived closed forms of delay and AoI approximately, demonstrating the tradeoff relationship between the two performance metrics. Chen *et al* in [18] have proposed an algorithm to solve the AoI-aware radio resource management problem in the scenario with Manhattan grid vehicle-to-vehicle network.

The above works are all devoted to information update issues. In practical scenarios, in addition to maintaining the cached information fresh, edge nodes also need to provision low-delay computation offloading services for mobile users. Both information updating and computation offloading processes consume considerable bandwidth, while the overall bandwidth of an edge node is limited. Thus, the computation offloading issues should also be taken into consideration.

Extensive works have investigated computation offloading problems, seeking to tradeoff the conflicts between sufficient computation capacities and additional wireless transmission requests when offloading. The works on computation offloading can be divided into centralized schemes and decentralized schemes with respect to the manners in which the computation offloading decisions are made. Centralized schemes have made computation offloading decisions for multiple mobile users [19], [20], [21], [22], [23]. Decentralized schemes have also been studied to make computation offloading decisions for the scenario with a single user [24], [25], [26], [27] and the scenario with multiple users [7], [28], [29], [30]. It has been proved that the computation offloading problem is NP-hard when making computation offloading decisions for multiple users in a centralized manner [7]. Game theory has been introduced to characterize the selfish property of users when making computation offloading decisions for multiple users. Chen *et al* in [7], [28] have first presented potential game based computation offloading strategies for mobile users in the scenario with a single edge node and multiple edge nodes, respectively. Li *et al* in [29] have studied decentralized task offloading strategies for applications with statistical QoS guarantee. Zheng *et al* [30] have solved the computation offloading problem with multiple users under dynamic environment.

Most of the above works on computation offloading mainly focus on improving users' quality of experience with respect to processing delay and energy consumption. However, for applications requiring fresh data, keeping fresh information at the edge requires frequent information update from sensors, which consumes additional wireless bandwidth and thus degrades the user quality of experience during computation offloading. Therefore, information update and computation offloading should be jointly optimized to improve user experience while ensuring freshness of information at the edge node.

III. System Model

We consider the scenario as shown in Fig. 1. There are a set of $\mathbb{N} = \{1, 2, ..., N\}$ mobile users with delay-sensitive and freshness-aware computation tasks to be processed. Processing these tasks needs fresh information maintained at edge nodes, which requires a set of $\mathbb{M} = \{1, 2, ..., M\}$ sensors to update the fetched data frequently. All the sensors and mobile users can be connected to a base station (i.e., edge node) which provides wireless coverage, storage and computation resources. Let B represent the overall uplink bandwidth allocated for this type of application. We consider that when there exists one sensor requiring to update its data, the information update channel with bandwidth B^{U} is reserved for information update. In this case, the computation offloading channel has the bandwidth $B^{\mathrm{O}} = B - B^{\mathrm{U}}$; Otherwise, $B^{\mathrm{O}} = B$.

We consider a set of $\mathbb{T} = \{1, 2, ..., T\}$ time slots. In each time slot, no more than one sensor updates its data via the information update channel. Denote by $\hat{s}(t) = (s_m(t))_{m=1}^{M}$ the channel allocation decision of sensors in the tth ($t \in \mathbb{T}$) time slot. Specifically, $s_m(t) = 1$ if the information update channel is allocated to sensor m at the tth time slot. Otherwise, $s_m(t) = 0$. When allocating the information update channel to sensors, the interference constraint should be ensured, i.e.,

$$\sum_{m=1}^{M} s_m(t) \le 1. \tag{1}$$

When the information update channel is allocated to sensor m, it generates fresh data and sends to the edge node with success probability of p_m. Let $\Omega_m(t)$ be the variable indicating whether the mobile data of sensor m is successfully updated. For each sensor m, there is

$$\mathbb{E}\{\Omega_m(t)\} = p_m \mathbb{E}\{s_m(t)\}. \tag{2}$$

Note that when no sensors update their mobile data (i.e., $s_m(t) = 0$, $\forall m \in \mathbb{M}$), all the wireless resources are dedicated to offloading computation tasks, i.e.,

$$B^{\mathrm{O}}(\hat{s}(t)) = \begin{cases} B & \sum_{m=1}^{M} s_m(t) = 0 \\ B - B^{\mathrm{U}} & \sum_{m=1}^{M} s_m(t) = 1 \end{cases} \tag{3}$$

Thus, the channel allocation decision $\hat{s}(t)$ affects the computation offloading cost of mobile users through the bandwidth as in Eq. (3). In the computation offloading channel, mobile users with computation tasks to offload share the computation offloading channel based on Shannon's theorem. As mobile users share the limited wireless bandwidth of the computation offloading channel, each mobile user needs to decide either to process the computation tasks on its own mobile device or to offload to the edge node according to its own interest. Let $\hat{x}(t) = (x_n(t))_{n=1}^{N}$ represent the offloading decision of mobile user n at the tth time slot: $x_n(t) = 1$, if mobile user n decides to offload the computation tasks to the edge node, otherwise, $x_n(t) = 0$. We consider that when processing mobile tasks locally on mobile devices, the information at these mobile devices can be updated timely with the edge node, as updating information of mobile devices consumes the downlink resources of the edge node, which usually has much higher bandwidth than the uplink.

A. Performance Metrics of Sensors

We use AoI to depict the freshness of mobile data received from sensors. Denote by $A_m(t)$ the AoI of sensor m in the tth time slot. In the tth time slot , when the information update channel is allocated to sensor m and the generated data is transmitted successfully to the edge node, the AoI of sensor m in the tth time slot is reduced to 1. If the edge node cannot receive the fresh data from sensor m, the AoI increases by 1 over the last time slot. By summarizing the above results, the AoI is recursively updated as follows,

$$A_m(t+1) = A_m(t) + 1 - \Omega_m(t) A_m(t). \tag{4}$$

As the frequency that each sensor successfully transmits the fresh information to the edge node is important for sensor networks, the successful transmission frequency of each sensor (i.e., throughput) should be lower-bounded. Thus, for each sensor $m \in \mathbb{M}$, there is

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \Omega_m(t) \ge \varphi_m. \tag{5}$$

Here, φ_m is the lower bound of sensor m.

B. Interests of Mobile Users

Mobile users pursue their own interests when making computation offloading decisions. In this paper, we take processing delay and energy consumption of mobile tasks as the metrics of user interests. Denote by $G_n \triangleq \langle \alpha_n, \beta_n \rangle$ the mobile task of user n to be processed, where α_n is the computation requirement of the task (in CPU cycles) and β_n is the transmitted data volume when offloading the task.

When processing the mobile task of user n ($n \in \mathbb{N}$) locally at the mobile device, i.e., $x_n(t) = 0$, the processing delay is

$$D_n^{\mathrm{local}}(t) = \frac{\alpha_n}{f_n}. \tag{6}$$

Here, f_n is the local computation capacity of mobile user n. The energy consumption of local processing is

$$E_n^{\mathrm{local}}(t) = P_n \cdot D_n^{\mathrm{local}}, \tag{7}$$

where P_n is the processing power at mobile device of user n.

When offloading the mobile task of user n ($n \in \mathbb{N}$) to the edge node, i.e., $x_n(t) = 1$, the processing delay consists of transmission delay in the wireless network and computation delay at the edge node. As users offloading tasks to the edge node share the computation offloading channel based on Shannon's theorem, the transmission rate of mobile user n is given as

$$\begin{aligned} &r_n(t) = \\ &B^{\mathrm{O}}(\hat{s}(t)) \cdot \log_2(1 + \frac{P_n^{\mathrm{trans}} g_n}{w_n + \sum\limits_{i \in \{\dot{n} | x_{\dot{n}}(t) = 1 \& \dot{n} \neq n\}} P_i^{\mathrm{trans}} g_i}). \end{aligned} \tag{8}$$

Here, P_n^{trans} is the transmission power of user n, g_n is the channel gain, and w_n is the background interference irrelevant to the offloaded tasks. Thus, the transmission delay is

$$D_n^{\mathrm{trans}}(t) = \frac{\beta_n}{r_n}. \tag{9}$$

The computation delay at the edge node is

$$D_n^{\mathrm{comp}}(t) = \frac{\alpha_n}{f_n^{\mathrm{edge}}}, \tag{10}$$

where f_n^{edge} is the edge capacity allocated to mobile user n. Therefore, the processing delay of user n is given as

$$D_n^{\mathrm{edge}}(t) = D_n^{\mathrm{trans}} + D_n^{\mathrm{comp}}. \tag{11}$$

The energy consumption when offloading to the edge node can be computed as

$$E_n^{\mathrm{edge}}(t) = P_n^{\mathrm{trans}} \cdot D_n^{\mathrm{trans}} + E_n^{\mathrm{tail}}, \tag{12}$$

where E_n^{tail} represents the energy consumed to maintain the wireless interface on and to scan wireless signals [31].

C. Problem Formulation

For each edge node, the sensors and mobile users compete for the communication resources of the base station to update the information and offload computation tasks, respectively. For the sensors, the objective is to maintain the data at the edge node fresh. For the mobile users, the objective is to optimize the cost with respect to the delay and energy consumption when processing computation tasks, i.e.,

$$\min_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{n=1}^{N} C_n(t), \tag{13}$$

Here, $C_n(t)$ is represented by a weighted sum of processing delay and energy consumption,

$$C_n(t) = \begin{cases} D_n^{\mathrm{local}}(t) + \gamma_n * E_n^{\mathrm{local}}(t), & x_n(t) = 0 \\ D_n^{\mathrm{edge}}(t) + \gamma_n * E_n^{\mathrm{edge}}(t), & x_n(t) = 1 \end{cases} \tag{14}$$

where γ_n is the weight parameter specified by mobile user n. This paper jointly optimizes the channel allocation decisions $\hat{s}(t)$ for sensors and computation offloading decisions $\hat{x}(t)$ for mobile users, such that the computation offloading cost of mobile users is optimized while ensuring the freshness requirement of sensors. Thus, the problem formulation is as follows,

$$\begin{aligned} \mathbf{P1} \quad & \min \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{n=1}^{N} C_n(t), \\ s.t. \quad \mathbf{C1} \quad & \lim_{T \to \infty} \frac{E\{A_m(T)\}}{T} = 0, \qquad \forall m \in \mathbb{M} \\ \mathbf{C2} \quad & \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \Omega_m(t) \geq \varphi_m. \qquad \forall m \in \mathbb{M} \end{aligned} \tag{15}$$

The objective is to minimize long-term average computation offloading cost of mobile users. **C1** constrains the freshness requirements of sensors. **C2** enforces the throughput of sensors lower-bounded.

IV. Problem Analysis and Algorithm Design

In this section, we analyze problem **P1** and design an algorithm to solve this problem based on Lyapunov techniques.

A. Problem Analysis

In problem **P1**, the objective is to minimize the long-term average computation offloading cost of mobile users. Remember that when no sensors update the data in one time slot, the wireless resources are dedicated to offloading computation tasks for mobile users, as in Eq. (3). Thus, both the channel allocation decision $\hat{s}(t)$ of sensors and the computation offloading decisions $\hat{x}(t)$ of mobile users have an effect on the interests of mobile users, i.e., $\sum\limits_{n=1}^{N} C_n(t)$. When the computation offloading channel has more wireless resources (i.e., no sensors update the data), the offloading costs of mobile users decrease and they are more likely to offload the computation tasks. The constraint **C1** represents the freshness requirement of sensors. According to the evolvement property of $A_m(t)$ in Eq. (4), the AoI $A_m(t)$ of the current time slot is determined by the AoI $A_m(t-1)$ and the channel allocation decision of the last time slot $\hat{s}(t-1)$ (as in Eq. (4)). The channel allocation decision of the current time slot $\hat{s}(t)$ has an influence on the time-average throughput of the future. Therefore, problem **P1** is a nonlinear integer dynamic optimization problem.

Define the virtual queue $Q_m(t)$ as

$$Q_m(t+1) = \max\{Q_m(t) + \varphi_m - \Omega_m(t), 0\}. \tag{16}$$

From the above definition, the virtual queue $Q_m(t)$ represents the accumulative throughput debt of sensor m ($m \in \mathbb{M}$) across t time slots. Here, $Q_m(t)$ is introduced because ensuring constraint **C2** does not necessarily mean that the throughput of each time slot should be larger than the threshold strictly. We just need to empower the sensor that has larger accumulated throughput debt $Q_m(t)$ with high priority to update its data.

Let $\Pi(t) = (A_m(t), Q_m(t))_{m=1}^{M}$ denote the system state at the beginning of time slot t. In problem **P1**, the channel allocation decision $\hat{s}(t)$ and system state $\Pi(t)$ jointly determine the AoI and the accumulated throughput debt of the $(t+1)$th time slot. That means that $\hat{s}(t)$ and $\Pi(t)$ have an effect on the future channel allocation decision to ensure the long-term average

AoI and throughput constraints in **P1**. Thus, problem **P1** is a dynamic optimization problem over $\hat{s}(t)$. By summarizing the above analysis, problem **P1** is a dynamic optimization problem over $\hat{s}(t)$. When $\hat{s}(t)$ ($t \in \mathbb{T}$) is given, problem **P1** is reduced to a static optimization problem over $\hat{x}(t)$ in each time slot. It has been proved that even the reduced static optimization problem over $\hat{x}(t)$ is a NP-hard problem [7]. An algorithm with reduced computation complexity is required to solve **P1**.

B. Algorithm Design

In this subsection, we design an algorithm to solve **P1** based on the above analysis. We first decompose the long-term dynamic optimization problem **P1** into a static optimization problem over $\hat{s}(t)$ in each time slot using Lyapunov optimization techniques. Then, we solve the static optimization problem in each time slot to jointly determine the channel allocation decision $\hat{s}(t)$ ($m \in \mathbb{M}$) and the computation offloading decision $\hat{x}(t)$ ($n \in \mathbb{N}$).

1) Problem Decomposition: The evolvement of $A_m(t)$ is given as Eq. (4), and $A_m(t)$ can be seen as the backlog of a queue. In each time slot t, the arrival rate of the queue $a_m(t) = 1$, and the serving rate $b_m(t) = A_m(t)\Omega_m(t)$. Thus, ensuring the constraint **C1** is equivalent to enforcing the mean rate stability of the queue.

From the definition of the virtual queue $Q_m(t)$ in Eq. (16), $Q_m(t)$ has the following property.

Theorem 1. *Enforcing the stability of $Q_m(t)$ can ensure the throughput constraint of sensor m in problem **P1***

Proof. Please refer to Appendix A. □

With Theorem 1 and the analysis on $A_m(t)$, it can be concluded that solving problem **P1** is equivalent to optimizing computation cost of mobile users while ensuring the stability of $Q_m(t)$ and $A_m(t)$, which can be solved by using Lyapunov optimization techniques.

Define the Lyapunov function $L(t)$ as

$$L(t) = \frac{1}{2}\sum_{m=1}^{M}[Q_m(t)]^2 + \frac{1}{2}\sum_{m=1}^{M}[A_m(t)]^2. \tag{17}$$

From the definition of $L(t)$, it is indicated that when $L(t)$ is small, both the accumulated throughput debt $Q_m(t)$ and information age $A_m(t)$ of all sensors are low. Thus, the throughput constraint of problem **P1** can be ensured and the AoI of sensors can be improved by keeping $L(t)$ small.

Define the Lyapunov drift $\Delta(t)$ as

$$\Delta(t) = \mathbb{E}\{L(t+1) - L(t)|\Pi(t)\} \tag{18}$$

We seek to keep $L(t)$ small by minimizing the Lyapunov drift $\Delta(t)$ in each time slot t. From the definition of Lyapunov function $L(t)$, the Lyapunov drift $\Delta(t)$ is given as:

$$\begin{aligned}\Delta(t) &= \mathbb{E}\{L(t+1) - L(t)|\Pi(t)\}\\ &=\frac{1}{2}\sum_{m=1}^{M}\mathbb{E}\{[Q_m(t{+}1)]^2 - [Q_m(t)]^2|\Pi(t)\}\\ &+\frac{1}{2}\sum_{m=1}^{M}\mathbb{E}\{[A_m(t{+}1)]^2 - [A_m(t)]^2|\Pi(t)\}\end{aligned} \tag{19}$$

From the recursive property of $Q_m(t)$ in Eq. (16), it can be derived that

$$\begin{aligned}&\mathbb{E}\{[Q_m(t{+}1)]^2 - [Q_m(t)]^2|\Pi(t)\}\\ &= \mathbb{E}\{[\max\{Q_m(t) + \varphi_m - \Omega_m(t), 0\}]^2 - [Q_m(t)]^2|\Pi(t)\}\\ &\le \mathbb{E}\{[Q_m(t) + \varphi_m - \Omega_m(t)]^2 - [Q_m(t)]^2|\Pi(t)\}\\ &= \mathbb{E}\{-2Q_m(t)\Omega_m(t) + 2\varphi_m Q_m(t) + [\varphi_m - \Omega_m(t)]^2|\Pi(t)\}\\ &\overset{(a)}{\le} -2Q_m(t)p_m\mathbb{E}\{s_m(t)|\Pi(t)\}+2\varphi_m Q_m(t) + 1.\end{aligned} \tag{20}$$

Here, (a) is derived based on Eq. (2), and $[\varphi_m - \Omega_m(t)]^2 \le 1$.

From the recursive property of $A_m(t)$ in Eq. (4), it can be derived that

$$\begin{aligned}&\mathbb{E}\{[A_m(t{+}1)]^2 - [A_m(t)]^2|\Pi(t)\}\\ &= \mathbb{E}\{[A_m(t{+}1)]^2 - [A_m(t)]^2|\Omega_m(t) = 1\}\mathrm{Pro}\{\Omega_\mathrm{m}(\mathrm{t}) = 1|\Pi(\mathrm{t})\}\\ &+ \mathbb{E}\{[A_m(t{+}1)]^2 - [A_m(t)]^2|\Omega_m(t) = 0\}\mathrm{Pro}\{\Omega_\mathrm{m}(\mathrm{t}) = 0|\Pi(t)\}\\ &\overset{(b)}{=}\{1 - [A_m(t)]^2\}p_m\mathbb{E}\{s_m(t)|\Pi(t)\}\\ &+\{[A_m(t) + 1]^2 - [A_m(t)]^2\}\{1 - p_m\mathbb{E}\{s_m(t)|\Pi(t)\}\}\\ &= -A_m(t)[A_m(t) + 2]p_m\mathbb{E}\{s_m(\mathrm{t})|\Pi(t)\} + [2A_m(t) + 1].\end{aligned} \tag{21}$$

Here, Pro$\{\cdot\}$ represents the probability of the event. (b) is derived as $\mathrm{Pro}\{\Omega_m(t) = 1|\Pi(t)\} = \mathbb{E}\{\Omega_m(\mathrm{t})|\Pi(t)\}$, and $\mathbb{E}\{\Omega_m\}$ is given as Eq. (2).

Substitute Eq. (20) and Eq. (21) into Eq. (19), and the Lyapunov drift can be given as

$$\begin{aligned}&\Delta(t) \le\\ &\sum_{m=1}^{M} -[Q_m(t) + \frac{1}{2}A_m(t)(A_m(t) + 2)]p_m E\{s_m(t)|\Pi(t)\}\\ &+ \sum_{m=1}^{M}[\varphi_m Q_m(t) + A_m(t) + 1]\end{aligned} \tag{22}$$

The analysis in [6] indicates that the smaller $\Delta(t)$ is, the more likely $Q_m(t)$ and $A_m(t)$ ($\forall m \in \mathbb{M}$) are stabilized. In problem **P1**, in addition to the queues we want to stabilize, there is also a "penalty process" $y(t)$ we want to minimize, which is defined as

$$y(t) = \sum_{n=1}^{N} C_n(t). \tag{23}$$

Thus, we turn to minimize $\Delta(t) + Vy(t)$ in each time slot t ($t \in \mathbb{T}$) based on the Lyapunov optimization techniques. Here, V is a weight constant for computation offloading cost. Following Eq. (22), $\Delta(t) + Vy(t)$ is constrained as

$$\begin{aligned}&\Delta(t) + Vy(t) \le\\ &\sum_{m=1}^{M} -[Q_m(t) + \frac{1}{2}A_m(t)(A_m(t) + 2)]p_m E\{s_m(t)|\Pi(t)\}\\ &+ \sum_{m=1}^{M}[\varphi_m Q_m(t) + A_m(t) + 1] + Vy(t)\end{aligned} \tag{24}$$

Instead of directly minimizing $\Delta(t) + Vy(t)$, we seek to design an algorithm to minimize the right-hand-side of Eq.

(24). Here, $\sum_{m=1}^{M}[\varphi_m Q_m(t)+A_m(t)+1]$ only relies on $\Pi(t)$ and is independent of the channel allocation decision $\hat{s}(t)$ and the computation offloading decision $\hat{x}(t)$. Thus, we just need to optimize problem **P2** in each time slot as

$$\textbf{P2} \min_{\langle\hat{s}(t),\hat{x}(t)\rangle} \Lambda(t)=\sum_{m=1}^{M}-Z_m(t)E\{s_m(t)|\Pi(t)\}+V\sum_{n=1}^{N}C_n(t), \tag{25}$$

where $Z_m(t)=[Q_m(t)+\frac{1}{2}A_m(t)(A_m(t)+2)]p_m$.

2) Solving the Static Optimization Problem: In problem **P2**, the channel allocation decision $\hat{s}(t)$ affects the computation offloading cost $C_n(t)$ ($n\in\mathbb{N}$) of mobile users as follows. When no sensors update the data, i.e., $\sum_{m=1}^{M}s_m(t)=0$, all communication resources are dedicated to offloading mobile tasks, $B^{\mathrm{O}}(\hat{s}(t))=B$. If there exists one sensor updating the data, the bandwidth of the computation offloading channel is $B^{\mathrm{O}}(\hat{s}(t))=B-B^{\mathrm{U}}$. Thus, **P2** can be analyzed by dividing into two cases:

i) No sensors update the data. In this case, $s_m(t)=0$ ($\forall m\in\mathbb{M}$), and problem **P2** is reduced to the computation offloading problem given $B^{\mathrm{O}}=B$ as

$$\begin{aligned}\textbf{P3} \ \min &\sum_{n=1}^{N}C_n(\hat{x}(t))\\ s.t.\ \ & x_n(t)\in\{0,1\}\ \ \forall n\in\mathbb{N}.\end{aligned} \tag{26}$$

ii) There exists one sensor updating the data. In this case, $B^{\mathrm{O}}=B-B^{\mathrm{U}}$. As the computation offloading cost $C_n(t)$ is independent of the channel allocation decision when B^{O} is given, we just need to optimize the channel allocation decision to minimize $\sum_{m=1}^{M}-Z_m(t)E\{s_m(t)|\Pi(t)\}$, and the computation offloading decision to minimize $\sum_{n=1}^{N}C_n(t)$, separately. For the former, we need to decide which sensor among the M sensors to update its data. As the objective is to minimize $\sum_{m=1}^{M}-Z_m(t)E\{s_m(t)|\Pi(t)\}$, we just need to choose the sensor m with the maximum $Z_m(t)$. For the latter, we just need to solve the computation offloading problem given $B^{\mathrm{O}}=B-B^{\mathrm{U}}$.

By summarizing the above two cases, the optimal channel allocation decision $\hat{s}(t)$ and computation offloading decision $\hat{x}(t)$ can be determined, and the details are summarized in Algorithm 1.

In algorithm 1, the main computation complexity comes from selecting the sensor $m^{'}$ to update its data (Step 8) and solving problem **P3** to obtain the computation offloading decision $\hat{x}(t)$ (Step 12) in each time slot. Selecting the sensor $m^{'}$ has $O(M)$ complexity. Based on the analysis of [7], solving problem **P3** centrally is NP-hard as the computation offloading problem in the scenario with an edge node can be changed into a bin-packing problem. The main concern is how to solve problem **P3** with reduced complexity.

Game-theoretic analysis on the computation offloading problem has been extensively investigated to reduce the complexity with decentralized solutions. In this work, we seek to design an algorithm to solve **P3** based on the game-theoretic analysis. The computation offloading problem **P3** has the following property.

Algorithm 1 Freshness-Aware Channel Allocation and Computation Offloading

```
Input: The throughput constraint $\varphi_m$, and the success probability $p_m$ of sensors ($m\in\mathbb{M}$).
    The computation task $G_n$, computation capacity $f_n$, computation power $p_n$, allocated edge capacity $f_n^{\mathrm{edge}}$, and transmission power $p_n^{\mathrm{trans}}$ of mobile users ($n\in\mathbb{N}$).
Output: The channel allocation decision $\hat{s}^*(t)$ and the computation offloading decision $\hat{x}^*(t)$.
1:  Let $s_m^0(t)=0$ for $\forall m\in\mathbb{M}$.
2:  Let $B^{\mathrm{O}}(\hat{s}^0(t))=B$.
3:  Solve problem P3 given $B^{\mathrm{O}}(\hat{s}^0(t))$, and obtain $\hat{x}^0(t)$ and the corresponding computation offloading cost $\sum_{n=1}^{N}C_n(\hat{x}^0(t))$.
4:  Let $\Lambda^0(t)=V\sum_{n=1}^{N}C_n(\hat{x}^0(t))$.
5:  Let $A_m(1)=1$, $Q_m(1)=0$.
6:  for each $t\in\mathbb{T}$ do
7:      Compute $Z_m(t)$ for $\forall m\in\mathbb{M}$ as
        $Z_m(t)=[Q_m(t)+\frac{1}{2}A_m(t)(A_m(t)+2)]p_m$.
8:      $m^{'}=\arg\max_{m\in M}\{Z_m(t)\}$.
9:      $s_{m^{'}}(t)=1$, $s_m(t)=0$ ($\forall m\neq m^{'}$).
10:     Let $B^{\mathrm{O}}(\hat{s}(t))=B-B^{\mathrm{U}}$.
11:     Solve problem P3 given $B^{\mathrm{O}}(\hat{s}(t))$, and obtain $\hat{x}^{'}(t)$ and the corresponding computation offloading cost $\sum_{n=1}^{N}C_n(\hat{x}^{'}(t))$.
12:     $\Lambda^{'}(t)=-Z_{m^{'}}(t)+V\sum_{n=1}^{N}C_n(\hat{x}^{'}(t))$.
13:     if $\Lambda^0(t)\leq\Lambda^{'}(t)$ then
14:         $\hat{s}^*(t)=\hat{s}^0(t)$, $\hat{x}^*(t)=\hat{x}^0(t)$.
15:     else
16:         $\hat{s}^*(t)=\hat{s}^{'}(t)$, $\hat{x}^*(t)=\hat{x}^{'}(t)$.
17:     end if
18:     Update $A_m(t)$ according to (4) for $\forall m\in\mathbb{M}$.
19:     Update $Q_m(t)$ according to (16) for $\forall m\in\mathbb{M}$.
20: end for
```

Theorem 2. *Problem **P3** has finite improvement property.*

Proof. Please refer to Appendix B. □

With Theorem 2, we can conclude that the computation offloading game can reach the unique Nash equilibrium after finite improvement iterations. By exploiting this property, we try to reduce the computation complexity (which can be represented in number of improvement iterations) through enlarging the improvement range of each iteration. Thus in each iteration, we select the mobile user with the highest improvement range to update its computation offloading decision, and the number of iterations can be significantly reduced. The main idea can be summarized in Algorithm 2.

Algorithm 2 Computation Offloading Decision with Reduced Complexity

```
1: Initialization: each mobile user $n$ initially chooses to execute the computation task locally at the mobile device, i.e., $x_n^0(t) = 0$.
2: for each iteration $i$ do
3:   Initialize the update user set $\Phi = \emptyset$.
4:   for each mobile user $n \in \mathbb{N}$ do
5:     Let $x_n = 1 - x_n^{i-1}(t)$.
6:     if $C_n(x_n, x_{-n}^{i-1}(t)) < C_n(x_n^{i-1}(t), x_{-n}^{i-1}(t))$ then
7:       Add mobile user $n$ into the update user set $\Phi$.
8:       Compute the improvement range of $n$ as
         $\eta_n = C_n(x_n^{i-1}(t), x_{-n}^{i-1}(t)) - C_n(x_n, x_{-n}^{i-1}(t))$.
9:     end if
10:  end for
11:  if $\Phi = \emptyset$ then
12:    Break.
13:  end if
14:  Select $n_* \in \Phi$ with the largest improvement range $\eta_n$ to update its computation offloading decision, $x_{n_*}^i(t) = 1 - x_{n_*}^{i-1}(t)$.
15:  Keep the computation offloading decisions of the other mobile users unchanged, i.e., $x_{-n_*}^i(t) = x_{-n_*}^{i-1}(t)$.
16: end for
```

Here, $x_{-n}(t)$ represents the computation offloading decisions of mobile users except n. Let I denote the number of improvement iterations in Algorithm 2. In each iteration i, obtaining the update user set (Step 4-Step 9) and selecting the user with the largest improvement range to update the computation offloading decision (Step 14-Step 15) dominate the complexity of algorithm 2, which both have $O(N)$ complexity. Thus, solving problem **P3** with algorithm 2 has $O(NI)$ complexity.

Based on the definition of ordinal potential game in (31), the improvement trends of mobile user cost $C_n(\hat{x})$ and the ordinal potential function $U(\hat{x})$ are identical. To further analyze the number of improvement iterations I, we turn to analyze the improvement process of the ordinal potential function. According to the potential function in (32), there is

$$U(\hat{x}) \leq \frac{1}{2}[\max_{n\in N}(P_n^{\text{trans}} g_n)]^2 N^2 + \max_{n\in N}(P_n^{\text{trans}} g_n) \max_{n\in N}(\delta_n) N. \tag{27}$$

When the improvement iteration ends, the potential function $U(\hat{x}^{\text{end}}) > 0$. Thus, the overall improvement range of the potential function in algorithm 2 will not exceed

$$\frac{1}{2}[\max_{n\in N}(P_n^{\text{trans}} g_n)]^2 N^2 + \max_{n\in N}(P_n^{\text{trans}} g_n) \max_{n\in N}(\delta_n) N.$$

When each mobile user n changes its computation offloading decision from x_n to $1 - x_n$, the improvement range of the potential function evolves as

$$\begin{aligned} &U(x_n, x_{-n}) - U(1 - x_n, x_{-n}) \\ &= P_n^{\text{trans}} g_n |\delta_n - \sum_{i\neq n} P_i^{\text{trans}} g_i x_i| > 0. \end{aligned} \tag{28}$$

In algorithm 2, we select the mobile user n with the largest improvement range η_n to update its computation offloading decision in each improvement iteration (Step 14). Correspondingly, the improvement range of the potential function in the improvement iteration is also the largest. By summarizing the above analysis, the overall improvement range of the potential function is upper-bounded, and the improvement range in each iteration is maximized. Therefore, the number of improvement iterations I can be significantly reduced. Remember the computation complexity analysis of algorithm 1, and the complexity can be given as $O(T * M + T * NI)$.

C. Discussion

In the above analysis, we assume that fixed bandwidth is reserved for the information update channel (i.e., B^U) and no more than one sensor updates the information in each time slot. Our proposed algorithm can be easily extended to the scenario that the information update channel has varying bandwidth and multiple sensors are allowed to update their information simultaneously. As the information update channel with more bandwidth can allow more sensors to update the information, we just need to match the reserved bandwidth to the number of concurrently updating sensors. When all the sensors are allowed to update, further increasing the bandwidth is no longer beneficial for reducing AoI of sensors at the edge node. Thus, we at most need to run the algorithm similar to Algorithm 1 (the main difference stems from Step 8) M times for the scenario with changing information update bandwidth. For example, if the information update channel with $B_U(t)$ bandwidth can allow W sensors to update the information in one time slot, we just need to choose the first W sensors with the largest $Z_m(t)$ to update the information (in Step 8 of Algorithm 1). The computation offloading decisions can still be determined by Algorithm 2 with the only difference that the bandwidth is changed to $B - B_U(t)$. In this case, the overall computation complexity is $O(T * M \log M + T * MNI)$.

V. Simulation Results

A. Simulation Setup

We simulate an edge node covering a 50m range area. In this area, there are a total of M sensors which can fetch fresh information and a set of N mobile users with freshness-aware and cost-sensitive computation tasks to process. The total wireless bandwidth $B = 100$ Mbps, and the reserved bandwidth for the information update channel is $B^U = 40$ Mbps. We consider $T = M * 10^3$ time slots. The successful transmission probability of sensors p_m is within [0.8, 1.0] ($m \in \mathbb{M}$). The lower bound of successful transmission frequency is $\varphi_m \in \frac{[0.75,1]}{M}$ units of information per time slot, which satisfies the necessary and sufficient condition for the feasibility [32]. We take object recognition as the example of computation tasks, requiring $\alpha_n = 1000$ Megacycles computation and $\beta_n = 600$ KB data to be transmitted when offloading the task [7]. The computation capacity of mobile devices $f_n \in [1.0, 2.0]$ GHz. The local processing power on mobile devices is $P_n = 10$ mW [24], [28]. The edge capacity allocated to each edge device $f_n^{\text{edge}} = 10$ GHz. The transmission power of mobile devices is $P_n^{\text{trans}} = 100$ mW, and the background interference $\omega_n = -100$ dBm [28]. The

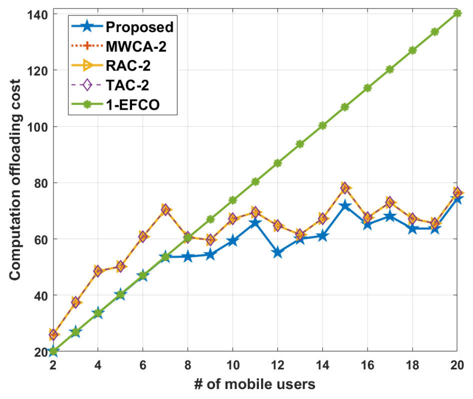


Fig. 2. Computation offloading cost with # of mobile users: number of sensors $M = 10$.

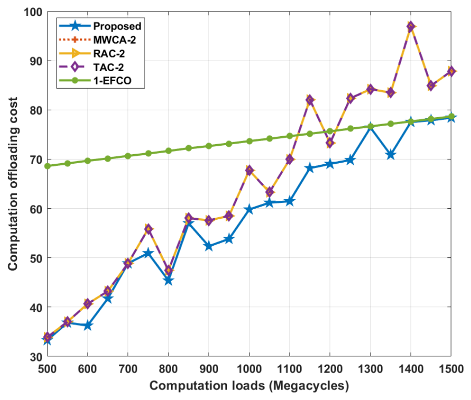


Fig. 4. Impact of computation loads on different algorithms: number of sensors $M = 10$, number of mobile users $N = 10$.

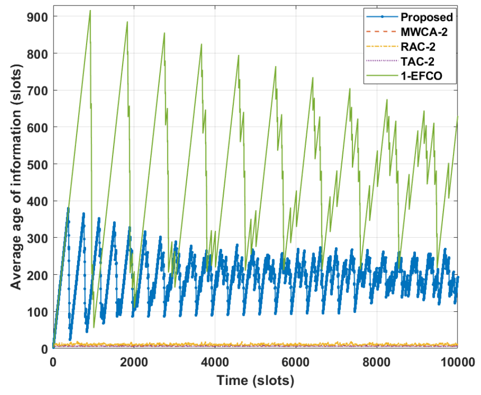


Fig. 3. AoI of different algorithms: number of sensors $M = 10$, number of mobile users $N = 10$.

channel gain of mobile devices g_n is set as d^{-o}, where d is the distance from edge node to mobile devices and o is the path loss parameter $o = 4$ [28].

We compare our proposed algorithm with four benchmark algorithms:

AoI-Prior algorithm (MWCA-2): In each time slot, the information upadate channel is reserved and allocated to the sensor with the largest weight $Z_m(t)$ which is similar as [14]. The computation offloading decision is made by algorithm 2.

Randomly-Allocate-Channel algorithm (RAC-2): In each time slot the information update channel is allocated to a sensor randomly. Mobile users make the computation offloading decision by algorithm 2.

Throughput-aware-Allocate-Channel algorithm (TAC-2): In each time slot, the information update channel is allocated to the sensor with the largest throughput debt. Mobile users make the computation offloading decision by algorithm 2.

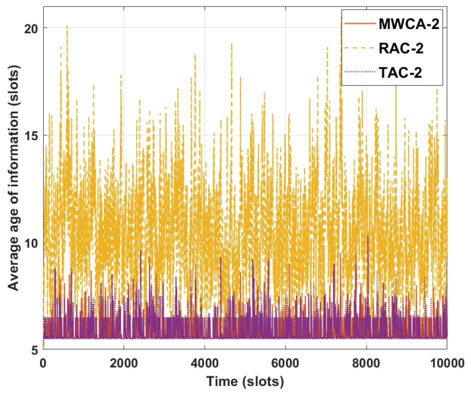
Edge-First-Computation-Offloading algorithm (1-EFCO): In each time slot, the channel allocation decision is made by algorithm 1. The computation tasks of mobile users are all offloaded to the edge for processing.

B. Simulation Results

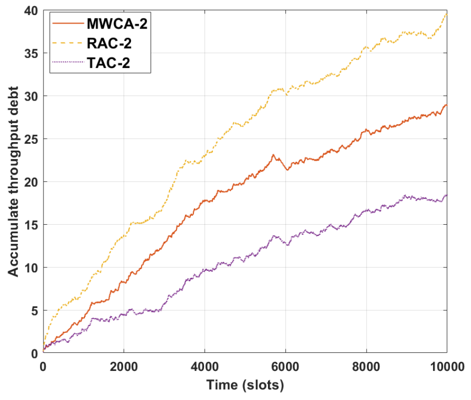
1) Computation Offloading Cost: The computation offloading cost of different algorithms are shown in Fig. 2. Compared to the benchmark algorithms, the proposed algorithm always yields the lowest computation offloading cost. When the number of mobile users are small, the proposed algorithm and the 1-EFCO algorithm yield the same and lower computation cost than the other three algorithms. That's because when the number of mobile users are small, the computation offloading channel can provide sufficient bandwidth, and all mobile users choose to offload the computation tasks to the edge node. Moreover, the computation offloading cost can be significantly reduced when more bandwidth is allocated to computation offloading, thus the proposed algorithm and the 1-EFCO algorithm devoted all bandwidth to computation offloading, yielding lower computation cost than the other three benchmark algorithms. When the number of mobile users increases, the bandwidth of the computation offloading channel gets scarce for all mobile user to offload computation tasks, and more mobile users choose to process the computation tasks on mobile devices. Increasing the bandwidth of the computation offloading channel cannot reduce the computation offloading cost apparently. Thus, the proposed algorithm has more approximate computation offloading cost with the other three benchmark algorithms when the number of mobile users is large.

2) Average AoI of Sensors: Fig. 3 shows the average AoI of sensors in different algorithms. The benchmark algorithms (apart from 1-EFCO) have lower AoI than our proposed algorithm, as in each time slot there exists a sensor selected to update its information without considering leaving the bandwidth for mobile users to offload computation tasks. The proposed algorithm can achieve much lower average AoI than 1-EFCO because 1-EFCO has much higher computation cost and only higher AoI can incent to reserve bandwidth for information updating of sensors.

3) Impact of Computation Loads: To evaluate the impact of computation loads on different algorithms, we conduct simulations and the results are shown in Fig. 4. It can be seen from Fig. 4 that the computation offloading costs of different algorithms overall increase with the computation loads of



(a) AoI of benchmark algorithms: number of sensors $M = 10$, number of mobile users $N = 10$.



(b) Accumulated throughput debt of benchmark algorithms: number of sensors $M = 10$, number of mobile users $N = 10$.

Fig. 5. Analysis of the proposed channel allocation policy

mobile users. The jitters stem from the randomness of the computation capacities of mobile devices and the wireless environment of mobile devices are in. Compared with the benchmark algorithms, the proposed algorithm always yields the minimum computation offloading cost. The computation offloading cost of the proposed algorithm increases more slowly with the computation loads than MWCA-2, RAC-2, and TAC-2, as more bandwidth is allocated for computation offloading when the computation loads are heavy while the other three algorithms have fixed bandwidth for computation offloading. When the computation loads are larger than 1300 Megacycles, all computation tasks are offloaded to the edge, thus the proposed algorithm has identical computation offloading cost with the 1-EFCO algorithm.

4) Further Analysis of the Channel Allocation Policy: To validate the efficiency of the proposed channel allocation policy (Step 8 of Algorithm 1), we compare the AoI and accumulated throughput debt of sensors as in Fig. 5. As in MWCA-2, RAC-2, and TAC-2, the computation offloading of mobile users is not considered, and MWCA-2 has the same channel allocation policy (i.e., choosing the sensor with the largest $Z_m(t)$ to update the information) with our proposed algorithm, the efficiency of the channel allocation policy can be validated by comparing the three benchmark algorithms.

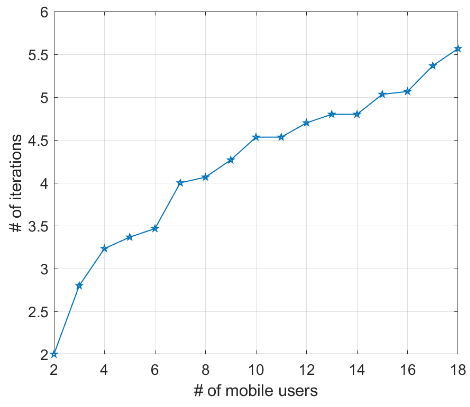


Fig. 6. Number of iterations with number of mobile users.

The simulation results demonstrate that the proposed channel allocation policy can yield the lowest AoI among the three algorithms and much lower accumulated throughput debt than RAC-2. TAC-2 has the lowest accumulated throughput debt as it always chooses the sensor with the largest accumulated throughput debt to update the information, which thus can reduce the overall accumulated throughput debt to the largest extent.

5) Scalability of the Computation Offloading Policy: According to the theoretical analysis in Sec. IV, Algorithm 2 can significantly reduce the number of iterations at the cost of centralized control over mobile users (which induces information collection overhead and additional computation complexity to choose the mobile user with the largest improvement range). In this section, we evaluate the scalability by simulation the number of iterations when achieving the equilibrium in Algorithm 2. The results are shown in Fig. 6. It is demonstrated that the number of iterations increases slower than linearly with the number of mobile users. Thus, Algorithm 2 has good scalability with the increasing number of mobile users.

VI. Conclusions

To our best knowledge, this paper is the first to jointly optimize the channel allocation and computation offloading decisions by considering the tradeoff between keeping fresh information at the edge node and reducing the computation offloading cost of mobile users. We formulate the problem as a nonlinear integer dynamic optimization problem, aiming at optimizing the computation offloading cost within the AoI and throughput constraints of sensors. To overcome the challenges of AoI-computation cost tradeoff, AoI time dependency and high complexity caused by heterogenous users, we propose an algorithm with reduced computation complexity based on Lyapunov optimization techniques and the potential game theory. Simulation results have demonstrated the effectiveness and scalability of the proposed algorithm. For the future work, the problem in the multi-edge scenario will be investigated and cooperation among edge nodes will be considered to provision fresh information and reduce computation offloading cost.

Appendix A
Proof of Theorem 1

From the definition of $Q(t)$ in (16), there is

$$Q_m(t+1) \geq Q_m(t) + \varphi_m - \Omega_m(t).$$

It can be derived that

$$\begin{aligned} Q_m(t+1) - Q_m(t) &\geq \varphi_m - \Omega_m(t) \\ Q_m(t) - Q_m(t-1) &\geq \varphi_m - \Omega_m(t-1) \\ &\vdots \\ Q_m(2) - Q_m(1) &\geq \varphi_m - \Omega_m(1). \end{aligned} \tag{29}$$

Sum up both sizes of (29) through T time slots, divide by T and take a limit, there is

$$\lim_{T\to\infty} \frac{Q_m(T+1) - Q_m(1)}{T} \geq \varphi_m - \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \Omega_m(t).$$

From the definition of queue rate stable, the stability of $Q(t)$ requires that

$$\lim_{T\to\infty} \frac{Q_m(T)}{T} = 0. \tag{30}$$

Moreover,

$$\lim_{T\to\infty} \frac{Q_m(1)}{T} = 0.$$

Therefore, enforcing the stability of $Q(t)$ is equivalent to ensuring

$$\varphi_m - \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \Omega_m(t) \leq 0.$$

Thus, the throughput constraint **C2** in problem **P1** can be satisfied.

Appendix B
Proof of Finite Improvement Property

Monderer *et al* in [8] have proved that an ordinal potential game has a unique Nash equilibrium and has finite improvement property. In this section, we first formulate the decentralized computation offloading decision problem as a computation offloading game. Thus, to prove that problem **P3** has finite improvement property, we just need to prove that the computation offloading game is an ordinal potential game. (The analysis of this section has removed the parameter t since problem **P3** is a static optimization problem).

We formulate the decentralized computation offloading problem as a computation offloading game. In this game, the players are the mobile users $\mathbb{N}$ with computation tasks to process.. The players' strategies are the computation offloading decisions $\hat{x}$ of mobile users. The players' costs are the computation offloading costs $C_n(\hat{x})$ $(n \in \mathbb{N})$ of mobile users (We denote local processing cost of user n as C_n^0). When multiple mobile users simultaneously offload computation tasks via the computation offloading channel, interference among the users can be caused as in Eq. (8). Hence the computation offloading cost of n can be denoted as $C_n(x_n, x_{-n})$. From the definition of the potential game [8], if for any player $n \in \mathbb{N}$ and any strategies of the other players x_{-n}, if there exists a function $U(\hat{x})$ satisfying

$$\begin{aligned} &C_n(x_n, x_{-n}) - C_n(x_n^{'}, x_{-n}) > 0 \\ &\text{iff} \;\; U(x_n, x_{-n}) - U(x_n^{'}, x_{-n}) > 0 \end{aligned} \tag{31}$$

for any x_n and $x_n^{'}$, the game is an ordinal potential game, and the function is an ordinal potential function. In the computation offloading game, we can find a function $U(\hat{x})$ given as

$$\begin{aligned} U(\hat{x}) = \frac{1}{2} \sum_{n=1}^{N} \sum_{l\neq n} P_n^{\text{trans}} g_n x_n P_l^{\text{trans}} g_l x_l \\ + \sum_{n=1}^{N} P_n^{\text{trans}} g_n \delta_n (1 - x_n), \end{aligned} \tag{32}$$

where $\delta_n = \frac{P_n^{\text{trans}} g_n}{2^{\frac{\beta_n (1+P_n^{\text{trans}}\gamma_n)}{B^O (C_n^0 - \frac{\alpha_n}{f_n^{\text{edge}}} - \gamma_n E_n^{\text{tail}})}} - 1} - \omega_n$. We can prove that $U(\hat{x})$ can satisfy (31) for any x_n, $x_n^{'}$ and x_{-n} $(\forall n \in \mathbb{N})$. Thus, the computation offloading game is an ordinal potential game, and we can solve problem **P3** by exploiting the finite improvement property of the game.

Acknowledgment

This work was supported by the National Key R&D Program of China (2018YFE0205503), the National Science Foundation of China (61902036; 61922017; 61921003), the Fundamental Research Funds for the Central Universities, the Open Research Fund of Key Laboratory of Space Utilization, Chinese Academy of Sciences(No. LSU-KFJJ-2019-03).

References

[1] "Mobile edge computing (mec); framework and reference architecture," White Paper, ETSI GS MEC 003 V1.1.1, Mar. 2016.

[2] B. Liang, "Mobile edge computing," in *Key Technologies for 5G Wireless Systems*, V. W. S. Wong, R. Schober, D. W. K. Ng, and L.-C. Wang, Eds. Cambridge University Press, 2017, pp. 76–91.

[3] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin, and X. S. Shen, "Cost-efficient resource provisioning for dynamic requests in cloud assisted mobile edge computing," *IEEE Transactions on Cloud Computing*, 2019.

[4] J. Zhong, R. D. Yates, and E. Soljanin, "Two freshness metrics for local cache refresh," in *IEEE International Symposium on Information Theory (ISIT'18)*. IEEE, 2018, pp. 1924–1928.

[5] C. Kam, S. Kompella, G. D. Nguyen, J. E. Wieselthier, and A. Ephremides, "Information freshness and popularity in mobile caching," in *IEEE International Symposium on Information Theory (ISIT'17)*. IEEE, 2017, pp. 136–140.

[6] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, Sep. 2010.

[7] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

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

[9] B. Zhou and W. Saad, "Minimum age of information in the internet of things with non-uniform status packet sizes," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 1933–1947, 2019.

[10] L. Corneo, C. Rohner, and P. Gunningberg, "Age of information-aware scheduling for timely and scalable internet of things applications," in *IEEE Conference on Computer Communications (INFOCOM'19)*. IEEE, 2019, pp. 2476–2484.

[11] Z. Qian, F. Wu, J. Pan, K. Srinivasan, and N. B. Shroff, "Minimizing age of information in multi-channel time-sensitive information update systems," in *IEEE Conference on Computer Communications (INFOCOM'20)*. IEEE, 2020, pp. 446–455.
[12] J. P. Champati, R. R. Avula, T. J. Oechtering, and J. Gross, "On the minimum achievable age of information for general service-time distributions," *arXiv preprint arXiv:2001.06831*, 2020.
[13] J. Lou, X. Yuan, S. Kompella, and N.-F. Tzeng, "Aoi and throughput tradeoffs in routing-aware multi-hop wireless networks," in *IEEE Conference on Computer Communications (INFOCOM'20)*. IEEE, 2020, pp. 476–485.
[14] I. Kadota, A. Sinha, and E. Modiano, "Optimizing age of information in wireless networks with throughput constraints," in *IEEE Conference on Computer Communications (INFOCOM'18)*. IEEE, 2018, pp. 1844–1852.
[15] Z. Jiang, S. Fu, S. Zhou, Z. Niu, S. Zhang, and S. Xu, "Ai-assisted low information latency wireless networking," *IEEE Wireless Communications*, vol. 27, no. 1, pp. 108–115, 2020.
[16] N. Lu, B. Ji, and B. Li, "Age-based scheduling: Improving data freshness for wireless real-time traffic," in *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2018, pp. 191–200.
[17] S. Zhang, L. Wang, H. Luo, X. Ma, and S. Zhou, "Aoi-delay tradeoff in mobile edge caching with freshness-aware content refreshing," *arXiv preprint arXiv:2002.05868*, 2020.
[18] X. Chen, C. Wu, T. Chen, H. Zhang, Z. Liu, Y. Zhang, and M. Bennis, "Age of information aware radio resource management in vehicular networks: A proactive deep reinforcement learning perspective," *IEEE Transactions on Wireless Communications*, vol. 19, no. 4, pp. 2268–2281, 2020.
[19] L. Yang, J. Cao, Z. Wang, and W. Wu, "Network aware mobile edge computation partitioning in multi-user environments," *IEEE Transactions on Services Computing*, 2018.
[20] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the internet of things with edge computing," *IEEE network*, vol. 32, no. 1, pp. 96–101, 2018.
[21] X. Ma, S. Zhang, W. Li, P. Zhang, C. Lin, and X. Shen, "Cost-efficient workload scheduling in cloud assisted mobile edge computing," in *Proc. IEEE/ACM International Symposium on Quality of Service (IWQoS'17)*, Vilanova i la Geltr, Spain, Jun 2017.
[22] J. Kwak, Y. Kim, J. Lee, and S. Chong, "Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 12, pp. 2510–2523, 2015.
[23] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1619–1632, 2018.
[24] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *Proc. IEEE International Conference on Computer Communications (INFOCOM'12)*, Orlando, Florida, USA, May 2012, pp. 2716–2720.
[25] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? the bandwidth and energy costs of mobile cloud computing," in *Proc. IEEE International Conference on Computer Communications (INFOCOM'13)*, 2013, pp. 1285–1293.
[26] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, 2012.
[27] R. Wolski, S. Gurun, C. Krintz, and D. Nurmi, "Using bandwidth data to make computation offloading decisions," in *Proc. IEEE International Parallel and Distributed Processing Symposium (IPDPS'08)*, Miami, Florida, USA, Apr. 2008, pp. 1–8.
[28] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2015.
[29] Q. Li, S. Wang, A. Zhou, X. Ma, A. X. Liu *et al.*, "Qos driven task offloading with statistical guarantee in mobile edge computing," *IEEE Transactions on Mobile Computing*, 2020.
[30] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Transactions on Mobile Computing*, vol. 18, no. 4, pp. 771–786, 2018.
[31] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. ACM, 2009, pp. 280–293.
[32] I.-H. Hou, V. Borkar, and P. Kumar, "A theory of qos for wireless," in *IEEE Conference on Computer Communications (IEEE'09)*, 2009, pp. 1–9.

Xiao Ma received her Ph.D. degree in Department of Computer Science and Technology from Tsinghua University and B.S. degree in Telecommunication Engineering from Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2018 and 2013. She is currently a lecturer at the State Key Laboratory of Networking and Switching Technology, BUPT. From October 2016 to April 2017, she visited the Department of Electrical and Computer Engineering, University of Waterloo, Canada. Her research interests include mobile cloud computing and mobile edge computing.

Ao Zhou received the P.H.D degrees in Beijing University of Posts and Telecommunications, Beijing, China, in 2015. She is currently an Associate Professor with State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. She has published 20+ research papers. She played a key role at many international conferences. Her research interests include Cloud Computing and Edge Computing.

Qibo Sun received the Ph.D. degree in communication and electronic system from the Beijing University of Posts and Telecommunication, China, in 2002. He is currently an Associate Professor with the Beijing University of Posts and Telecommunications. His research interests include services computing, the Internet of Things, and network security. He is a member of the China Computer Federation.

Shangguang Wang received his PhD degree at Beijing University of Posts and Telecommunications in 2011. He is Professor and Vice-director at the State Key Laboratory of Networking and Switching Technology (BUPT). He has published more than 150 papers, and played a key role at many international conferences, such as general chair and PC chair. His research interests include service computing, cloud computing, and mobile edge computing. He is a senior member of the IEEE, and the Editor-in-Chief of the International Journal of Web Science.