IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 16, 2019, accepted May 9, 2019, date of publication May 30, 2019, date of current version June 13, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2919651

Stochastic Computation Offloading and Scheduling Based on Mobile Edge Computing

XIAO ZHENG[1,2], MINGCHU LI[1,2], MUHAMMAD TAHIR[1,2], YUANFANG CHEN[3], AND MUHAMMAD ALAM[3]

[1]School of Software, Dalian University of Technology, Dalian 116620, China
[2]Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian 116620, China
[3]School of Cyberspace, Hangzhou Dianzi University, Hangzhou 310018, China

Corresponding author: Yuanfang Chen (chenyuanfang@hdu.edu.cn)

This work was supported in part by the National Nature Science Foundation of China under Grant 61572095, Grant 61877007, and Grant 61802097, and in part by the Project of Qianjiang Talent under Grant QJD1802020.

ABSTRACT To improve the quality of service (QoS) for mobile users (MUs) and the quality of experience (QoE) of mobile devices (MDs), mobile edge computing (MEC) is a promising approach that offloads a part of the computing task from MDs to nearby MUs. In this paper, we study computation offloading involving multiple users and multiple base stations (BSs), where the MD that is connected to the MU is wirelessly charged and BSs are available to be selected for computation offloading. We model the process of solving an optimal computation offloading policy into a Markov decision process (MDP), in which our goal is to maximize the long-term utility performance. Therefore, a computation offloading policy is obtained based on the energy queue state, the task queue state, and the channel states between the MUs and BSs. To address the problem of high dimensionality in the state space, we decompose the MDP into a series of single-agent MDPs with reduced state spaces and apply an online local learning algorithm to learn the optimal state value functions. Inspired by the structure of the utility function, we propose an algorithm based on combining Q-function reconstruction with the post-decision state. It is proved that the proposed algorithm can converge to an optimal computation offloading policy. The experimental results show that our algorithm achieves significant performance in computation offloading and schedule compared with the other three basic policies.

INDEX TERMS Quality-of-service (QoS), quality of experience (QoE), mobile device (MD), mobile edge computing (MEC), Markov decision process (MDP), post-decision state.

I. INTRODUCTION

The rapid development of the Internet of Things (IoT) has brought about exponentially growing types of terminal connections and mobile data traffic and more diverse service scenarios [1]. Low latency, high energy efficiency, high reliability and high density connectivity have become essential requirements for future mobile communication systems [2], also accelerated the development and implementation of the fifth generation (5G) mobile communication system. Compared to the fourth generation (4G), the greatest technological shift lies in the objective of its service, which changes from the communication between people to an objective of a comprehensive connection between things. 5G is an extension of 4G, which is just another phase in the evolution of existing wireless communication technologies, where more emphasis is placed on user experience. In the 5G era, mobile internet services, in addition to mobile phones and tablets, have added many new business scenarios, such as autonomous driving, virtual reality, augmented reality, and cloud desktops. 5G can also be applied to real-life IoT scenarios, such as intelligence smart grids, smart agriculture, smart cities, and environmental monitoring [3]. The emergence of these new services places greater demands on the delay, energy efficiency and reliability, etc. Needless to say, the existing 4G communication technology cannot meet the low-latency, high-efficiency and high-reliability application requirements [4]. In the context of the above applications, the academic community has proposed a corresponding solution, which is to use mobile edge computing (MEC) for computing and storage [5]. MEC is an

The associate editor coordinating the review of this manuscript and approving it for publication was Zakirul Alam Bhuiyan.

innovative computing mode in addition to centralized cloud computing, which deploys computing, storage, caching, etc. at the edge of the network and closer to the user, regardless of geographical or network distance. This strategy not only reduces the wasted resources of the traditional cloud computing back-haul link, greatly reduces the delay. In addition, this strategy meets the expanding computing requirements of terminal devices and ensures high reliability when processing tasks. In modern years, more computationally intensive applications have emerged and become popular. However, MDs are often limited by resources such as the battery capacity and local CPU computing power. Therefore, when performing computationally intensive tasks on MDs, the quality of experience (QoE) of the computing tasks can be greatly affected by the limited processing power of the device and ultimately fail to meet the quality-of-service (QoS) requirements of MUs [6]. By offloading a portion of the application tasks from MDs to nearby MEC, a promising alternative method, which reduces the computational task execution time and may inevitably prolong the lifetime of the MDs can be provided. This paper mainly considers computation offloading at an MEC that is composed of a set of MDs, and each MD is wirelessly charged. Recently, significant work has been conducted on the design of computation offloading decision-making. Such research is mainly based on a one-time optimization, which fails with respect to long-term computational offloading performance. For example, in [7], Wang et al. proposed a direct multiplicative alternative algorithm to maximize revenues by optimizing the computation offloading decision, content cashing policy and resource allocation in an infinite energy multi-user MEC system with the computational latency constrained. In [8], Wang et al. used Lagrangian duality to minimize the overall system energy consumption. Hu et al. in [9] designed a two-stage method based on joint energy and time allocation when considering cooperative computational offloading in an infinite energy assisted transmission MEC system.

For a multi-user MEC system, computation offloading requires the wireless transmission of data. Designing a wireless radio resource between an MD and the MEC over a common wireless radio access network (WRAN) is quite a challenge. Many computation offloading heuristics should be considered, including but not limited to the channel state over time, the task arrival and the energy state of the MD in the existing dynamic environment [10]. Liu et al. [11] proposed an optimal computation offloading delay problem for an MDP and designed an efficient one-dimensional search algorithm to find the optimal solution. However, this method relies on the state of the previous channel in addition to the task arrival status. Mao et al. in [12] used the Lyapunov algorithm to form an optimal computation offloading policy in the MEC system for an MD with wireless energy harvesting. Meanwhile, the same method is used to consider the energy delay trade-off in the case of computation task offloading by Jiang and Mao [13] and Liu et al. [14]. Lyapunov optimization can only achieve a near optimal solution, and so Xu et al. [15] proposed a reinforcement learning (RL) [16], [17] algorithm to optimize the computation offloading such that knowledge of the previous network states is not required. However, [15] only considered the computation offloading of a set of MDs in a BS, failed to consider user movements, and did not guarantee random offloading. In addition, Chen et al. in [18] and [19] respectively proposed a scheme for multi-user computation offloading using the game algorithm and the Lyapunov algorithm. These were all based on mobile edge cloud computing and were static.

When the MEC encounters an ultra-dense network or a DQN, multiple BSs (with different data transmission states) are selected to offload a given computation resource. In this case, the explosion in the state space makes conventional reinforcement learning algorithms infeasible. Moreover, wireless charging is integrated into the MEC system to obtain sustainable computing performance, which makes the design of the stochastic computation offloading policy more challenging. This paper proposes a stochastic computation offloading policy combining the Q-valued function with the post-decision station.

The main contributions of this paper are as follows:

(1) We propose the stochastic computation offloading policy in the WRAN in which the time-varying computational resources and communication qualities are taken into account.

(2) To address the problem of the state space, we resort to an online local learning algorithm based on combining the Q-valued iterative function with a post-decision state to find an optimal computation offloading policy.

(3) The decision process of each MU is modelled as a single-agent Markov decision process (MDP). Then, we propose a linear decomposition method for each-MU MDP to reduce the computational complexity of each MU.

(4) Finally, we use experimental results to validate the theoretical study of this prove that the proposed algorithm outperforms three baseline schemes.

The rest of this paper is organized as follows. Section 2 introduces the system models and problem formulation. Section 3 describes the problem of designing an optimal computation offloading decision policy as an MDP and proposes the optimal decision policy. Section 4 presents the specific details of the online local learning algorithm. To verify the proposed solution, we use experiments to validate our solution in different situations considering two aspects in Section 5. The conclusions of the paper are highlighted in Section 6. The notations of this paper are summarized in Table 1.

II. SYSTEM MODEL

In this section, as shown in Figure 1:

We shall consider a virtualized WRAN containing a set of BSs and a set of MDs. Both MEC services and traditional communication transmission services are supported by the public physical network infrastructure. It is also assumed that the MD is wirelessly charged, the energy that is obtained is

TABLE 1. Key expressions used in this paper.

W	bandwidth allocated to the MEC
$B/\mathcal{B}$	BSs set
$\mathcal{A}$	collection of joint decision policies
$X/\mathcal{X}$	set of network states
g_b^i	channel gain state between the MU and BS b
q^i	MU's task queue status
$q^{(max)}$	maximum task queue length
$p^{(max)}$	maximum transmit power
δ	duration of one decision slot
μ	input data size of a task
ν	required CPU cycles for a task
$f^{(max)}$	maximum CPU-cycle frequency
a^i	arrival status of the task
q_e^i	MU's energy queue status
$q_{(e)}^{(max)}$	maximum energy queue length
χ/χ^i	MU's network status
Ψ	decision policies of the MU
s/s^i	MU's task offloading decision
e/e^i	MU's energy unit allocation
a_e/a_e^i	energy unit arrivals
m/m^i	MU-BS association state
V	state-value function
Q/Q^i	state-action value function
u	utility function of the MU
$1_{(\Omega)}$	indicator function that equals 1 if the condition Ω is satisfied, and otherwise 0.
r^i	achievable data rate of users
h^i	handover time
p_{tr}^{max}	maximum transmit power
t_{tr}^i	task transmit time
t^i	task execution time
ζ	Signal processing time

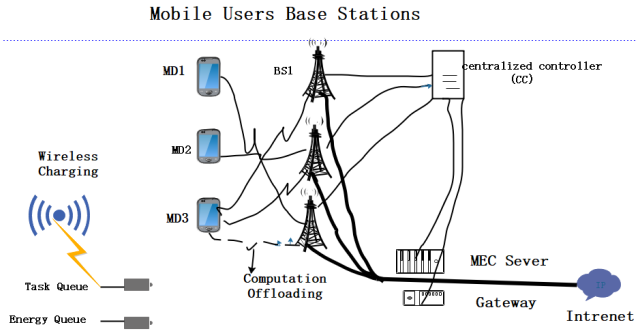

FIGURE 1. Diagram depicting Mobile-Edge Computing (MEC) in a wireless radio access network (WRAN).

stored in an energy queue, and a centralized control (CC) unit is responsible for all control decisions in the WRAN. MEC is placed at the edge of the network to provide rich resources for the MUs. The MU strategically offloads a computation task to the MEC via a BS. The case in which an MU may suddenly leave while offloading its computation task is also considered. The goal of this paper is to design an optimal computation offloading decision policy. The time is divided into equal-length time slots δ (in seconds) and is indexed by an integer $i \in N$.

The computation offloading task from the user across the time slot is an independent identical distribution of a Bernoulli random variable with a parameter $\lambda \in [0, 1]$. The size of the computing task for each user is (μ, ν), where μ denotes the input data size (bits), and ν denotes the total number of CPU cycles required to complete the computing task. A computation task in the queue can be either scheduled on the local MD or offloaded it to a remote MEC server. At the beginning of each decision time slot i, an MU makes a joint decision policy (s^i, e^i), where $s^i \in \{0 \cup \mathcal{B}\}$ represents the computation offloading decision policy, and $e^i \in N$ is the amount of energy to be allocated. If the user performs the local computation on their own device, namely, $s^i > 0$. When the MU chooses to offload the computation task to the MEC service via a BS, then $s^i \in \mathcal{B}$ and $s^i = 0$. Note that when $e^i = 0$, no task will be executed in the queue.

When an MU executes the local computation at a MD during a decision time slot i, i.e., $s^i = 0$. Let f^i be the computation capability (i.e., CPU cycles per second) of the MU. Moreover, the CPU-cycle frequency is constrained by $f^i \leq f_{CPU}^{max}$. The time required for local computation is as follows:

$$t_{mobile}^i = \frac{\nu}{f^i}. \tag{1}$$

where ν represents the total number of CPU cycles required to complete the computation task.

We can compute the achievable data rate of the user who chooses to offload computation tasks to the MEC via a BS:

$$r^i = W \cdot log_2(1 + \frac{g_b^i \cdot p_{tr}^i}{E}). \tag{2}$$

Let g_b^i be the channel gain between the MU and the BS $b \in \mathcal{B}$ during the decision time slot i, E is the received average power of the interference plus the additive background Gaussian noise, and

$$p_{tr}^i = \frac{e^i}{t_{mobile}^i}, \tag{3}$$

is the transmittance power where

$$t_{tr}^i = \frac{\mu}{r^i}. \tag{4}$$

is the time that the task input data is transmitted. In addition, e^i and μ indicate energy unit allocation by the MU and the input data size, and t_{mobile}^i and r^i represent the number of CPU cycles required to complete a computing task and the achievable data rate of users, respectively.

When an MU chooses to offload the computing task to an MEC, i.e, $s^i \in \mathcal{B}$. The association between the MU and BS has to be established for the first time. We assume that the user processes a computing task locally or no task is executed at the decision time slot $i-1$. Then, the association between the user and the BS will not occur, namely, $m^i = m^{i-1}$. In this case, no handover will be triggered. let $m^i \in \mathcal{B}$ be the system status between the user and the BS during the decision time slot.

$$m^i = b \cdot 1_{(s^{(i-1)}=b, b\in\mathcal{B})} + b \cdot 1_{((s^{(i-1)}=0)\wedge(m^{(i-1)}=b))}. \tag{5}$$

The channel state transmission is modelled as a finite state discrete Markov chain. Moreover, when the MU randomly moves, the MU can choose different BSs so that a handover time between the two BSs will occur. Let the handover time be

$$h^i = \zeta \cdot 1_{(s^i \in \mathcal{B}) \wedge (s^i \neq m^i)}, \tag{6}$$

where ζ (in seconds) is the time for processing the signal when one handover happens.

In addition, we assume in this paper that the battery capacity at the MD of the MU is limited and the received energy units across the time horizon take integer values. Let q_e^i be the length of the user energy queue at the beginning of the decision time slot i, which evolves according to

$$q_e^{i+1} = min\left\{q_e^i - e^i + a_e^i, q_e^{max}\right\}, \tag{7}$$

where $q_e^{max} \in \mathbb{N}$ denotes the battery capacity limit, and $a_e^i \in \mathbb{N}$ denotes the number of energy units received by the end of the decision time slot i.

III. PROBLEM FORMULATION

In this section, we study the problem of effective computation offloading and the optimal solution in the MDP framework.

An important indicator for evaluating the computation offloading task is the time delay experience. A computing task's execution time can be seen as the period when the task arrives in the task queue until it is processed and leaves the task queue. Thus, the experienced time includes the task queue time and the data processing time. Therefore, we obtain the task execution time as follows:

$$t^i = \begin{cases} t^i_{mobile}, & \text{if } e^i > 0 \quad \text{and } s^i = 0; \\ h^i + t^i_{tr}, & \text{if } e^i > 0 \quad \text{and } s^i \in \mathcal{B}; \\ 0, & \text{if } e^i = 0. \end{cases} \tag{8}$$

When the queue is full during the decision time slot, i.e, $0 < t^i \leq \delta$, the computing task will be discarded and ensure that the queue will not flow.

We denote a computing task discard as follows:

$$\rho^i = max\left\{q^i - r^i + a^i - q^{max}, 0\right\}, \tag{9}$$

where r^i is the number of tasks that were transferred at the decision time slot i.

When the computing task can be processed locally during the decision time slot in the queue, the dynamics of the computation task queue of the MU can be defined as follows:

$$\sigma^i = \left\{q^i - 1_{(0<t^i\leq\delta)} + a^i, q^{max}\right\}. \tag{10}$$

When a computation task remains in the queue for a decision time slot, a delay will be incurred by the task, that is, δ seconds. Let the queuing delay during the decision time slot i equal the length of a task queue, i.e.,

$$\eta^i = q_e^i - 1_{\{t^i>0\}} \tag{11}$$

If the MU moves randomly during the computation offloading time slot, it will generate the payment for the local execution and computation offloading at the MEC service remote. This payment is denoted as follows:

$$\upsilon^i = \alpha(min\left\{t^i, \delta\right\}) \cdot 1_{(s^i=0)} + \beta(min\left\{t^i, \delta\right\} - h^i) \cdot 1_{(s^i\in\mathcal{B})}, \tag{12}$$

where $\alpha \in \mathbb{R}_+$ represents the price that is paid for the local computation in each time unit, and $\beta \in \mathbb{R}_+$ represents that for the MEC service in each time unit.

The network state of the MU during each decision time slot i can be expressed as $\boldsymbol{\chi}^i = (m^i, g^i, q^i) \in \mathcal{X} = \{0, ..., \mathcal{B}\} \times \{\times_{b\in\mathcal{B}}\mathcal{G}_b\} \times \{0, 1, .., q^{max}\}$, where $g^i = (g_b^i : b \in \mathcal{B})$. At the beginning of time slot i, the MU strategically selects a joint task offloading action $(s^i, e^i) \in \mathcal{A} = \{(0) \cup \mathcal{B}\} \times \left\{0, 1, ..., Q_{(e)}\right\}$.

We define an immediate utility at the decision time slot i to measure the task computation qualities for the MU as,

$$u(\boldsymbol{\chi}^i, (s^i, e^i)) = \alpha_1 u^{(1)}(\sigma^i) + \alpha_2 u^{(2)}(\eta^i) + \alpha_3 u^{(3)}(\upsilon^i) + \alpha_4 u^{(4)}(\rho^i), \tag{13}$$

where we suppose that $u^{(1)}(\cdot)$, $u^{(2)}(\cdot)$, and $u^{(3)}(\cdot)$ are positive monotonically deceasing functions, $u^{(1)}(\cdot)$, $u^{(2)}(\cdot)$, $u^{(3)}(\cdot)$ and $u^{(4)}(\cdot)$ measure the satisfactions of the task execution delay, the task queuing delay, the payment of accessing the MEC service and the computation task drops, respectively; and $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \in \mathcal{R}_+$ are the wights that combine the different types of functions with the different performances into a common utility function. To unify the notation expression, we rewrite $u^{(1)}(\cdot)$, $u^{(2)}(\cdot)$, $u^{(3)}(\cdot)$ and $u^{(4)}(\cdot)$ as $u_1(\boldsymbol{\chi}^i, (s^i, e^i))$, $u_2(\boldsymbol{\chi}^i, (s^i, e^i))$,$u_3(\boldsymbol{\chi}^i, (s^i, e^i))$ and $u_4(\boldsymbol{\chi}^i, (s^i, e^i))$.

Definition 1: A joint task computation offloading and energy allocation decision policy $\boldsymbol{\Psi}$ is expressed as the mapping function: $\boldsymbol{\Psi} : \mathcal{X} \to \mathcal{A}$. Namely, $\boldsymbol{\Psi}$ is the mapping from a user state to an action. Moreover, the MU determines the joint decision action $\boldsymbol{\Psi}(\boldsymbol{\chi}^i) = (\boldsymbol{\Psi}_{(s)}(\boldsymbol{\chi}^i), \boldsymbol{\Psi}_{(e)}(\boldsymbol{\chi}^i)) = (s^i, e^i) \in \mathcal{A}$ according to $\boldsymbol{\Psi}$ after considering the network state $\boldsymbol{\chi} \in \mathcal{X}$ during the beginning of each decision time slot i.

Given a joint decision policy $\boldsymbol{\Psi}$, the $\left\{\boldsymbol{\chi}^i : i \in \mathcal{N}_+\right\}$ represents a controlled Markov chain with the following state transmission distribution probability. The joint decision policy $\boldsymbol{\Psi}$ induces a probability distribution over the sequence of global network states $\left\{\boldsymbol{\chi}^i : i \in \mathcal{N}_+\right\}$, and hence a probability distribution over the sequences of the per-time slot utility $\{u(\boldsymbol{\chi}^i, (s^i, e^i)) : i \in \mathcal{N}_+\}$. From the assumptions on the system states, the channel states and the energy states, $\boldsymbol{\chi}^i, i \in \mathcal{N}_+$, is Markovian with the following transition probability:

$$Pr\left\{\boldsymbol{\chi}^{i+1}|\boldsymbol{\chi}^i, \boldsymbol{\Psi}(\boldsymbol{\chi}^i)\right\} = Pr\left\{m^{i+1}|m^i, \boldsymbol{\Psi}(\boldsymbol{\chi}^i)\right\} \cdot Pr\left\{q^{i+1}|q^i, \boldsymbol{\Psi}(\boldsymbol{\chi}^i)\right\} \cdot \prod_{b\in\mathcal{B}} Pr\left\{g_b^{i+1}|g_b^i\right\}, \tag{14}$$

where $Pr\{.\}$ represents the probability of a user's action, $\boldsymbol{\Psi}(\boldsymbol{\chi}^i) = (s^i, e^i) \in \mathcal{A}$ represents the global channel state,

and s^i and e^i represent the joint task offloading and energy allocation policies, respectively.

Taking the expectation with respect to the sequence of each time slot utility, the expected long-term utility of an MU for a given initial state $(\boldsymbol{\chi})^1 = \boldsymbol{\chi}$ can be characterized as follows:

$$V(\boldsymbol{\chi}, \boldsymbol{\Psi}) = E_{\boldsymbol{\Psi}}[(1-\gamma) \cdot \sum_{i=1}^{\infty} (\gamma)^{i-1} \cdot \mu(\boldsymbol{\chi}^i, \boldsymbol{\Psi}(\boldsymbol{\chi}^i)) | \boldsymbol{\chi}], \quad (15)$$

where $\boldsymbol{\chi} = (s, q, \mathcal{G}) \in \mathcal{X}$, $\mathcal{G} = (g_b, b \in \mathcal{B})$, $\gamma \in [0, 1)$ denotes the discount factor, and $(\gamma)^{i-1}$ denotes the discount factor to the $(i-1)$-th power. $\mu(\boldsymbol{\chi}^i, \boldsymbol{\Psi}(\boldsymbol{\chi}^i))$ is the per-slot utility over the sequence of network states $\boldsymbol{\chi}^i$ under action $\boldsymbol{\Psi}(\boldsymbol{\chi}^i)$ at decision time slot i. $V(\boldsymbol{\chi}, \boldsymbol{\Psi})$ is also called the state-value function for the user in the state $\boldsymbol{\chi}$ under $\boldsymbol{\Psi}$.

Problem 1: The purpose of the user is to design an optimal decision policy $\boldsymbol{\Psi}^*$ that maximizes the expected long-term utility, $V(\boldsymbol{\chi}, \boldsymbol{\Psi})$, for any given initial network state $\boldsymbol{\chi}$, which can be formed by in the next

$$\boldsymbol{\Psi}^* = argmax_{\boldsymbol{\Psi}} V(\boldsymbol{\chi}, \boldsymbol{\Psi}), \quad \forall \boldsymbol{\chi} \in \mathcal{X}. \quad (16)$$

$V(\boldsymbol{\chi}) = V(\boldsymbol{\chi}, \boldsymbol{\Psi}^*)$ is called the optimal state-value function, $\forall \boldsymbol{\chi} \in \mathcal{X}$.

The decision policy attaining the optimal state-value function can be expressed as solving the following Bellman's equation [20]: $\forall \boldsymbol{\chi} \in \mathcal{X}$,

$$V(\boldsymbol{\chi}) = \max_{(s,e)} \Big\{ (1-\gamma) u(\boldsymbol{\chi}, (s, e)) + \gamma \sum_{\boldsymbol{\chi}'} Pr\left\{\boldsymbol{\chi}' | \boldsymbol{\chi}, (s, e)\right\} V(\boldsymbol{\chi}') \Big\}, \quad (17)$$

where the $u(\boldsymbol{\chi}, (s, e))$ represents the achieved utility when a joint decision policy $(s, e) \in \mathcal{Y}$ is executed under network state $\boldsymbol{\chi}$, and $\boldsymbol{\chi}' = (q', s', g_b') \in \mathcal{X}$ represents the next network state with $g' = (g_b' : b \in \mathcal{B})$.

We express the right-hand side of equation (16) as

$$Q(\boldsymbol{\chi}, (s, e)) = \Big\{ (1-\gamma) \cdot u(\boldsymbol{\chi}, (s, e)) + \gamma \cdot \sum_{\boldsymbol{\chi}'} Pr\left\{\boldsymbol{\chi}' | \boldsymbol{\chi}, (s, e)\right\} \cdot V(\boldsymbol{\chi}') \Big\} \quad (18)$$

$\forall \boldsymbol{\chi} \in \mathcal{X}$.

The optimal state-value function $V(\boldsymbol{\chi})$ can be directly attained from

$$V(\boldsymbol{\chi}) = \max_{(s,e)} Q(\boldsymbol{\chi}, (s, e)). \quad (19)$$

By replacing equation (18) with equation (17), we then obtain

$$Q(\boldsymbol{\chi}, (s, e)) = (1-\gamma) \cdot u(\boldsymbol{\chi}, (s, e)) + \gamma \cdot \sum_{\boldsymbol{\chi}'} Pr\left\{\boldsymbol{\chi}' | \boldsymbol{\chi}, (s, e)\right\} \cdot \max_{(s', e')} Q(\boldsymbol{\chi}', (s', e')), \quad (20)$$

where we let $(s', e') \in \mathcal{A}$ be a joint decision policy carried out under the network state $\boldsymbol{\chi}'$. In fact, the computation task's arrival and the amount of energy that can be obtained by the end of decision time slot are not available in advance.

The solutions of equation (16) are generally based on the policy iteration or value iteration where it is necessary to fully know the channel state transfer information, the computation task's arrival status and the obtained energy status. It is a great challenge to not know the previous channel transmission states and data arrivals. To address the challenge, we define an online local learning algorithm based on combination of the Q-valued function reconstruction with a post-decision [21], [22] to learn the optimal decision policy. Here, the newly arrived data are independent of the channel state and the computation offloading decision policy.

IV. FINDING THE OPTIMAL POLICY

In this section, we will address the optimal computation offloading decision policy by combining the Q-valued function reconstruction with the post-decision state.

At the current decision time slot, the post-decision state of an MU is defined by $\tilde{\boldsymbol{\chi}}^i = (\tilde{s}^i, \tilde{g}^i, \tilde{q}^i - r^i) \in \mathcal{X}$, where $\tilde{g}^i = g^i$ and $\tilde{\boldsymbol{\Psi}}(\tilde{\boldsymbol{\chi}}^i) = \boldsymbol{\Psi}(\boldsymbol{\chi}^i) - \boldsymbol{\Psi}^*(s^i, e^i)$.

Motivated by an additive structure, we can linearly decompose the utility function in (12) into two parts, corresponding to $\alpha_1 u^{(1)}(\cdot) + \alpha_2 u^{(2)}(\cdot) + \alpha_3 u^{(3)}(\cdot)$ and $\alpha_4 u^{(4)}(\cdot)$. The state transmission probability from $\boldsymbol{\chi}^i$ to $(\boldsymbol{\chi}^i)'$ can be expressed as follows:

$$\begin{aligned} &Pr\left\{(\boldsymbol{\chi}^i)' | \boldsymbol{\chi}^i, \boldsymbol{\Psi}(\boldsymbol{\chi}^i)\right\} \\ &= Pr\left\{(\boldsymbol{\chi}^i)' | \tilde{\boldsymbol{\chi}}^i\right\} Pr\left\{\tilde{\boldsymbol{\chi}}^i | \boldsymbol{\chi}^i, \boldsymbol{\Psi}(\boldsymbol{\chi}^i)\right\} \\ &= Pr\left\{\tilde{g}^i\right\} Pr\left\{\boldsymbol{\Psi}'(\boldsymbol{\chi}^i) - \tilde{\boldsymbol{\Psi}}(\boldsymbol{\chi}^i)\right\}, \end{aligned} \quad (21)$$

where $Pr\left\{\tilde{\boldsymbol{\chi}}^i | \boldsymbol{\chi}^i, \boldsymbol{\Psi}(\boldsymbol{\chi}^i)\right\} = 1$. Let $\tilde{V}(\tilde{\boldsymbol{\chi}}^i)$ be the each-MU optimal post-decision state value function given by:

$$\tilde{V}(\tilde{\boldsymbol{\chi}}^i) = (1-\gamma) u_4(\boldsymbol{\chi}^i, (s^i, e^i)) + \gamma \cdot \sum_{(\boldsymbol{\chi}^i)' \in \mathcal{X}} Pr\left\{(\boldsymbol{\chi}^i)' | \boldsymbol{\chi}^i\right\} V(\boldsymbol{\chi}^i)'. \quad (22)$$

At the beginning of the computation offloading decision time slot i, because the data arriving later are independent of the previous state, we only consider the last utility, i.e., the amount of data to be transmitted, R^i, during scheduling time slot i is expressed as follows:

$$R^i = \max_{(s,e)} \Big\{ (1-\gamma)(\alpha_1 u^{(1)}(\boldsymbol{\chi}^i, (s^i, e^i)) + \alpha_2 u^{(2)}(\boldsymbol{\chi}^i, (s^i, e^i)) + \alpha_2 u^{(3)}(\boldsymbol{\chi}^i, (s^i, e^i)) + \tilde{V}(\tilde{\boldsymbol{\chi}}^i) \Big\}. \quad (23)$$

Combining equations (12) and (16), we can get

$$V(\boldsymbol{\chi}^i) = \max_{(s,e)} \Big\{ (1-\gamma)(\alpha_1 u^{(1)}(\sigma^i) + \alpha_2 u^{(1)}(\eta^i) + \alpha_3 u^{(3)}(\upsilon^i)) + \tilde{V}(\tilde{\boldsymbol{\chi}}^i) \Big\}. \quad (24)$$

Through equation (12), we find that the optimal state value function of each user can be directly attained from the user's optimal post-decision state value function that maximizes all feasible computation offloading policies.

To the best of our knowledge, due to the random movement of the user and at the end of the scheduling decision time slot, new arrival data are not available in advance, and so the data are discarded at this time. In this case, we do not directly compute the user's optimal post-decision state value function that is represented by (21). This approach is based on the investigation of the local network state $\boldsymbol{\chi}^i$, and the number of packet departures r^i, the number of packet arrivals a^i, the number of packet drops $max\left\{q^i - r^i + a^i - q^{max}, 0\right\}$, at current scheduling slot i. We propose an online local learning algorithm to find the optimal post-decision state value function by reconstructing the value iterative equation. At the next decision time slot $i+1$, we dynamically update the user's post-decision state value function as follows:

$$\tilde{V}(\tilde{\boldsymbol{\chi}}^{i+1}) = (1-\zeta^i)\tilde{V}(\tilde{\boldsymbol{\chi}}^i) + \zeta^t((1-\gamma)u^{(4)}(\boldsymbol{\chi}^i,(s^i,e^i)) + \gamma V(\boldsymbol{\chi}^{i+1}), \quad (25)$$

where $\zeta^i \in [0,1)$ denotes the learning rate, and the convergence of the learning method is guaranteed by $\sum_{i=1}^{\infty}\zeta^i = \infty$ and $\sum_{i=1}^{\infty}(\zeta^i)^2 < \infty$. Therefore, that at the next decision time slot $i+1$ is evaluated by the following:

$$V(\boldsymbol{\chi}^{i+1}) = \max_{(s^i,e^i)}\Big\{(1-\gamma)(\alpha_1 u^{(1)}(\boldsymbol{\chi}^{i+1},(s^{i+1},e^{i+1})) + \alpha_2 u^{(2)}(\boldsymbol{\chi}^{i+1},(s^{i+1},e^{i+1})) + \alpha_3 u^{(3)}(\boldsymbol{\chi}^{i+1},(s^{i+1},e^{i+1})) + \tilde{V}(\tilde{\boldsymbol{\chi}}^{i+1})\Big\}. \quad (26)$$

An online local learning algorithm for finding the optimal post-decision state value function of the user is summarized as Algorithm 1. Theorem 1 ensures the convergence of the online local learning algorithm.

Theorem 1: For each user, the state value function $\left\{\tilde{V}(\tilde{\boldsymbol{\chi}}^i) : \forall i \in \mathcal{N}_+\right\}$ generated by Algorithm 1 converges to the optimal post-decision state value function $\tilde{V}(\tilde{\boldsymbol{\chi}}^i)$, $\forall\tilde{\boldsymbol{\chi}} \in \mathcal{X}$, if and only if the learning rate ζ^i is satisfied $\sum_{i=1}^{\infty}\zeta^i = \infty$ and $\sum_{t=1}^{\infty}(\zeta^i)^2 < \infty$.

The proof of Theorem 1 is given in Appendix.

Algorithm 1 An Online Local Learning Algorithm for Finding the User's Optimal Post-Decision State Value Function at the Decision Time Slot i in the Network

1: **initialize** The user's post-decision state value function, $\tilde{V}(\tilde{\boldsymbol{\chi}}^i), \forall\tilde{\boldsymbol{\chi}}^i \in \mathcal{X}$.
2: **repeat**
3: The user chooses network state $\boldsymbol{\chi}^i$, computes $\tilde{V}(\tilde{\boldsymbol{\chi}}^i)$ according to (21) and $V(\boldsymbol{\chi}^i)$ according to (23) at the beginning of each scheduling decision time slot i.
4: According to (22), the user makes computation offloading scheduling decisions.
5: After the data is scheduled to be transmitted, the user observes the post-decision state $\tilde{\boldsymbol{\chi}}^i = (s^i, g^i, q^i - r^i)$ and the utility $u^{(4)}(\boldsymbol{\chi}^i,(s^i,e^i))$ realized by data discarding at scheduling decision time slot i. At the next scheduling decision time slot $i+1$, the network state is $\boldsymbol{\chi}^{i+1} = (\tilde{q}^i + a^i, g^{i+1})$
6: The user computes the state value function $V(\boldsymbol{\chi}^{i+1})$ and updates the post-decision state value function $\tilde{V}(\tilde{\boldsymbol{\chi}}^{i+1})$ according to equations (25) and (24), respectively.
7: The scheduling decision time slot is updated by $i \leftarrow i+1$.
8:**until** Our predefined stop conditions are met.

V. NUMERICAL RESULTS

In this section, we process to evaluate the computation offloading performance obtained from our proposed online local learning algorithm. When this scheme is executed, Algorithm 1 determines the user's optimal post-decision state value function within the scheduling decision time slot.

A. GENERAL SETUP

We suppose the channel between the MU and the BS is Rayleigh channel across the scheduling time slots. Meanwhile, suppose exist $B = 4$ BSs connecting the user with the MEC server in the system. At each decision time slot i, the channel state across the base band is an exponentially distributed random variable with a mean of $\bar{H}$(dB). The value of H^i is determined by the common channel state space of discrete values, i.e., $\mathcal{H} = \{-16,-14,-12,-10,-8,-6,-4\}$(dB), the amount of energy from wireless transmission obeys the Poisson arrival process, and the average arrival rate is defined as $\lambda_{(e)}$ (data per scheduling decision time slot).

while the $u^{(1)}(\boldsymbol{\chi}^i,(s^i,e^i))$, $u^{(2)}(\boldsymbol{\chi}^i,(s^i,e^i))$, $u^{(3)}(\boldsymbol{\chi}^i,(s^i,e^i))$, $u^{(4)}(\boldsymbol{\chi}^i,(s^i,e^i))$ in (12) were chosen as the exponential functions. namely,

$$u_1(\boldsymbol{\chi}^i,(s^i,e^i)) = \alpha\cdot u^{(1)}(\boldsymbol{\chi}^i,(s^i,e^i)) = \alpha_1\cdot e^{-\sigma^i} \quad (27)$$
$$u_2(\boldsymbol{\chi}^i,(s^i,e^i)) = \alpha\cdot u^{(2)}(\boldsymbol{\chi}^i,(s^i,e^i)) = \alpha_2\cdot e^{-\eta^i} \quad (28)$$
$$u_3(\boldsymbol{\chi}^i,(s^i,e^i)) = \alpha\cdot u^{(3)}(\boldsymbol{\chi}^i,(s^i,e^i)) = \alpha_3\cdot e^{-\upsilon^i} \quad (29)$$
$$u_4(\boldsymbol{\chi}^i,(s^i,e^i)) = \alpha\cdot u^{(4)}(\boldsymbol{\chi}^i,(s^i,e^i)) = \alpha_4\cdot e^{-\rho^i} \quad (30)$$

This experiment is completed by an Q-valued iterative function learning model, other general parameters used in the simulation experiments are given in Table 2.

For comparison with performance, we also simulated three basic experiments:

1) Mobile Execution-The user performs all computation offloading tasks on the local MD with the maximum possible energy, i.e., at each decision time slot i, $s^i = 0$ and

$$e^i = \begin{cases} min\left\{q^i, \lfloor(f_{(CPU)}^{(max)})^3\cdot\tau\rfloor\right\} & \text{if } q^i > 0; \\ 0, & \text{if } q^i = 0; \end{cases} \quad (31)$$

where the allocation of energy considers the maximum CPU-cycle frequency $f_{(CPU)}^{(max)}$ and $\lfloor\cdot\rfloor$ denotes the floor function.

TABLE 2. Parameter values in experiments.

Parameter	Value
Channel bandwidth W	10^6 Hz
Decision slot duration δ	10^{-3} second
Noise power E	10^{-3} Watt
Input data size μ	10^3 bits
CPU cycle V	600 cycles per bit
Maximum CPU-cycle frequency $f_{(CPU)}^{(max)}$	2×10^9 Hz
Maximum transmit power $p^{(max)}$	2 Watt
Handover time ζ	2×10^{-3} second
Maximum energy queue length $q^{(max)}$	4 units
Maximum data task queue length $q^{(max)}$	4 tasks
Weights $\alpha_1, \alpha_2, \alpha_3$	3,9,5
Discount factor γ	0.9
Exploration probability e	0.01
Delay interval τ	10^{-28}

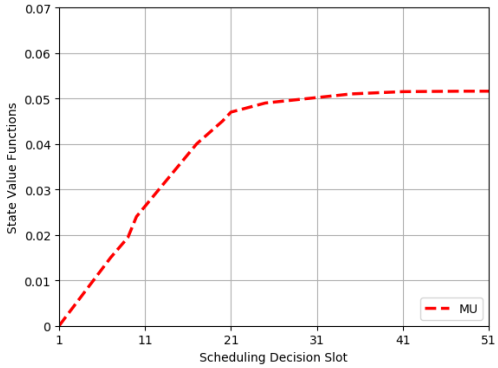

FIGURE 2. Diagram convergence property of our proposed online learning algorithm.

2) Server Execution-Since the maximum possible energy queue in the energy queue meets the maximum transmission energy limit, the user always chooses a BS that gets the minimum execution delay to perform the offload scheduling computation task size for the MEC sever.

3) Greedy Execution-At each decision time slot, the user decides to perform a computing task on the local device or offload it to the MEC for processing with the goal of minimizing the immediate task execution delay.

B. EXPERIMENT RESULTS

We conduct experiments under different conditions to verify the proposed scheme in this paper.

1) EXPERIMENT 1-CONVERGENCE OF THE PROPOSED ALGORITHM

In this experiment, our main purpose is to verify the convergence of our proposed algorithm, i.e, online local learning algorithm based on post-decision state. We suppose that there are totally 10 channels shared by the user in the wireless radio network. At each scheduling decision time slot, each user's task is achieved according to an average arrival rate $\lambda = 2$ (data per scheduling decision time slot) with a Poisson distribution. We have a fixed channel state mean $\bar{H} = -2$ dB for all users. We draw the simulation variables in $V(\boldsymbol{\chi}, (s, e))$ (where $\boldsymbol{\chi} = (4, 4, 4, (-16, -16, -12, -10, -16, -16))$, $(s, e) = (2, 4)$) for proposed algorithm at the decision time slot in Fig. 2, which shows that our proposed algorithm converges at a reasonable speed.

2) EXPERIMENT 2-TASK ARRIVAL RATE PERFORMANCE

This experiment tries to prove the average computation offloading performance per scheduling decision time slot in terms of the average utility, the average execution delay, the average task discards and the average MEC service payment under different computing task arrival probability. The results are showed in Fig.3. Fig.3(a) illustrates the average utility performance when the MU implements local learning algorithm. Figs.3(b)-(d) illustrate the average task execution delay, the average task drops, the average MEC service payment.

Each graph represents the performance of the proposed solution and the comparison of the other three basic computation offloading performance. We can see that the proposed scheme has achieved huge gains in terms of average utility. As the probability of computing tasks arrival increases, the average utility decreases due to the increase in average task execution delay, average task discards and average MEC service payments. Although each user's average execution delay and average task discard can be compared with other three schemes during an average decision time slot. Since there is not enough energy in the energy queue, to avoid task discarding, only part of the queue energy is used to allocate processing scheduling tasks, thus resulting in more queue tasks. This reason is explained by the network setting that changes the average arrival rate in the simulation. As the average task arrival rate increases, the average revenue performance decreases. Because there are not enough channels for all users during one decision time slot, there might be the case that only some of channel be scheduled for execution, which gives rise to the number of tasks being discarded.

3) Experiment 3-Performance under Various Means of Channel States:

We compare the user's performance from the proposed scheme to the baselines below:

(1) Channel-sensitive decision policy-The user estimates the channel requirements required for task transmission based on channel state rather than queue state at each decision time slot.

(2) Queue-sensitive decision policy-At each decision time slot, the user determines performance in getting a channel or not considering maximizing the expected long-term number of tasks being delivered [23].

(3) Random decision policy-This policy randomly generates a value with one channel or no for the user at a decision time slot, then submits it to the network with maximum performance. This means that the random policy in the network is not considered dynamically.

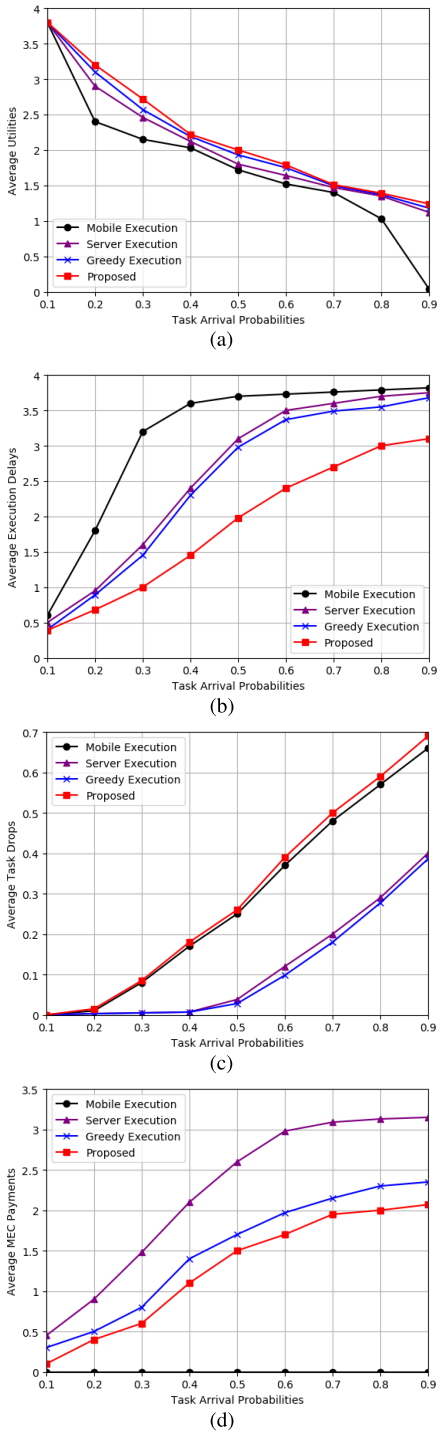

FIGURE 3. Average computation offloading performance in contrast with task arrival probabilities. (a) Average utility per decision time slot. (b) Average execution delay per decision time slot. (c) Average task discard per decision time slot. (d) Average MEC service payment per decision time slot.

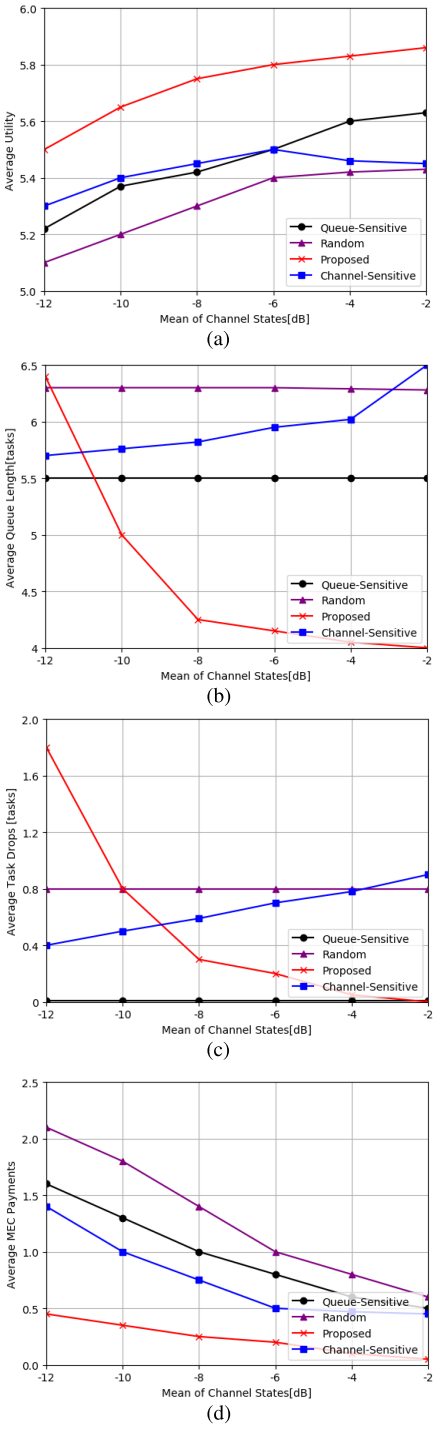

FIGURE 4. Average computation offloading performance in contrast with means of channel states. (a) Average utility per decision time slot. (b) Average execution delay per decision time slot. (c) Average task discard per decision time slot. (d) Average MEC service payment per decision time slot.

We do simulation experiments to compare the performance of each user obtained from the proposed scheme with other three baselines in term of the channel mean. The parameter values we used in the simulation are as follows: $\bar{H} = -6$ dB, and $\lambda = 3$ (task each decision time slot), average execution delay, average task discard, average the MEC service payments as well as each user's average utility through the entire learning process are depicted in Figure 4. From the figure we can clearly see that the better the average channel condition, the less the average transmission energy is necessary for data transmission. This is because the number of channel

required is guaranteed to meet user needs. However, baseline method 1 only considers the channel state without considering the number of task queues, so each user's average task discard and average revenue achieve the worst performance.

Overall, from Experiments 2 and Experiments 3, the proposed local learning based scheme outperforms the three baseline schemes with respect to the per-MU average utility performance in all simulations. When making the channel status and task scheduling decisions following our proposed scheme, the BSs and the MUs obviously strike a strategic tradeoff between the immediate payoff performance and the potential payoffs from future competitive interactions.

VI. CONCLUSION

In this paper, we study the design of stochastic computing offload policy in a MEC system, which considers the energy obtained from the wireless network, the arrival of computing tasks, and the channel state dynamically generated from time-varying change between users and BSs. The computation offloading problem is modeled as a MDP problem, in which an online local learning algorithm is proposed to find the optimal computation offloading policy. This algorithm overcomes the high dimensionality of the state space that does not require any previous dynamic information network. Compared with other three basic schemes, our proposed scheme can achieve better long-term utility performance, indicating an optimal trade-off between the computing task execution delay, task discard and MEC service payment.

APPENDIX

PROOF OF THEOREM 1

Due to the symmetry during the learning process, we consider the online local learning algorithm based on the optimal post-decision state value function for an arbitrary user. Let $Y : \mathbb{R}^{|\mathcal{X}|} \to \mathbb{R}^{|\mathcal{X}|}$ be a mapping such that

$Y(\tilde{V}) = [Y(\tilde{V}(\tilde{\boldsymbol{\chi}}^i) : \tilde{\boldsymbol{\chi}}^i \in \mathcal{X}]$ where $\tilde{V} = [\tilde{V}(\tilde{\boldsymbol{\chi}}^i) : \tilde{\boldsymbol{\chi}}^i \in \mathcal{X}]$ and $Y(\tilde{V}(\tilde{\boldsymbol{\chi}}^i))$ is defined by:

$$\begin{aligned} &Y(\tilde{V}(\tilde{\boldsymbol{\chi}}^i)) \\ &= (1-\gamma)u^{(3)}(\boldsymbol{\chi}^i, (s^i, e^i)) + \gamma \sum_{(\boldsymbol{\chi}^i)' \in \mathcal{X}} pr\left\{((\boldsymbol{\chi})^i)' | \tilde{\boldsymbol{\chi}}^i\right\} \\ &\quad \times \max_{(s^i, e^i)} \Big\{(1-\gamma)(\alpha_1 u^{(1)}(((\boldsymbol{\chi})^i)', (s^i, e^i)) \\ &\quad + \alpha_2 u^{(2)}(((\boldsymbol{\chi})^i)', (s^i, e^i))\tilde{V}((\boldsymbol{\chi})^i)'\Big\} - \tilde{V}(\tilde{\boldsymbol{\chi}}^i) \end{aligned} \tag{32}$$

Define $Z : \mathbb{R}^{|\mathcal{X}|} \to \mathbb{R}^{|\mathcal{X}|}$ by:

$$Z(\tilde{V}) = Y(\tilde{V}) + \tilde{V} \tag{33}$$

The updating rule for the post-decision state value functions of MU n given by (24) at current decision slot i can be reformulated as the recursion:

$$\tilde{V}(\tilde{\boldsymbol{\chi}}^{i+1}) = \tilde{V}(\tilde{\boldsymbol{\chi}}^i) + \zeta^i(Z(\tilde{V}(\tilde{\boldsymbol{\chi}}^i))) - \tilde{V}(\tilde{\boldsymbol{\chi}}^i) + \xi(\tilde{\boldsymbol{\chi}}^i) \tag{34}$$

where $\xi(\tilde{\boldsymbol{\chi}}^i)$ is the martingale, given by:

$$\begin{aligned} \xi(\tilde{\boldsymbol{\chi}}^i) = \gamma[&\max_{(s',e')} \Big\{(1-\gamma)(\alpha_1 u^{(1)}((\boldsymbol{\chi}^i)', (s', e')) \\ &+ (\alpha_2 u^{(2)}((\boldsymbol{\chi}^i)', (s', e')) + \tilde{V}(\boldsymbol{\chi}^i)'\Big\} \\ &- \sum_{(\boldsymbol{\chi}^i)' \in \mathcal{X}} Pr\left\{(\boldsymbol{\chi}^i)' | \tilde{\boldsymbol{\chi}}\right\} \\ &\times \max_{(s',e')} \Big\{(1-\gamma)(\alpha_1 u^{(1)}((\boldsymbol{\chi}^i)', (s', e')) \\ &+ (\alpha_2 u^{(2)}((\boldsymbol{\chi}^i)', (s', e')) + \tilde{V}(\boldsymbol{\chi}^i)'\Big\}] \end{aligned} \tag{35}$$

Let $\mathcal{F}(i) = \sigma(\left\{V^{\bar{i}}, \xi^{\bar{i}} : \bar{i} \le i\right\})$ be the history of the learning procedure during the first i decision slots, From(33), it is automatic that

$$E[\xi(\tilde{\boldsymbol{\chi}}^i)|\mathcal{F}(i)] = 0 \tag{36}$$

By taking the conditional variance of both sides of (33), it is not hard to obtain as follow, where Var$(\cdot)$ denotes the variance.

$$\begin{aligned} E[(\xi(\tilde{\boldsymbol{\chi}}^i))^2|\mathcal{F}(i)] \le \gamma\, &Var((1-\gamma)(\alpha_1 u^{(1)}((\boldsymbol{\chi}^i)', (s', e')) \\ &+ \alpha_2 u^{(2)}((\boldsymbol{\chi}^i)', (s', e'))) + \gamma \max_{\tilde{\boldsymbol{\chi}}^i \in \mathcal{X}} (\tilde{V}(\tilde{\boldsymbol{\chi}}^i))^2 \end{aligned} \tag{37}$$

From a lemma in [24], the convergence of the learning rule in (32) then is equivalent to the convergence of the associated ordinary differential equation,i.e.,

$$\dot{\tilde{V}} = Z(\tilde{V}) - \tilde{V} := Y(\tilde{V}). \tag{38}$$

For a discount factor $\gamma \in [0, 1)$, (31) yields

$$\begin{aligned} &\left\|Z(\tilde{V}) - Z(\tilde{V}')\right\|_\infty \\ &= \max_{\boldsymbol{\chi} \in \mathcal{X}} |Z(\tilde{V}(\boldsymbol{\chi})) - Z(\tilde{V}'(\boldsymbol{\chi}))| \le \gamma \left\|\tilde{V} - \tilde{V}'\right\|_\infty \forall \tilde{V}, \tilde{V}'. \end{aligned} \tag{39}$$

In particular, Z is a norm γ-contraction mapping. The Assumptions 1,2,3 and 5 of [25] hold. Therefore, Theorem 1 establishes the convergence of the learning process.

REFERENCES

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

[2] B. P. Rimal, D. P. Van, and M. Maier, "Mobile edge computing empowered fiber-wireless access networks in the 5G era," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 192–200, Feb. 2017.

[3] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Netw. Appl.*, vol. 18, no. 1, pp. 129–140, Feb. 2013.

[4] M. R. Rahimi, J. Ren, C. H. Liu, A. V. Vasilakos, and N. Venkatasubramanian, "Mobile cloud computing: A survey, state of art and future directions," *Mobile Netw. Appl.*, vol. 19, no. 2, pp. 133–143, 2014.

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

[6] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[7] 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.
[8] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, Mar. 2018.
[9] X. Hu, K.-K. Wong, and K. Yang, "Wireless powered cooperation-assisted mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 17, no. 4, pp. 2375–2388, Apr. 2018.
[10] Y. Chen, M. Guizani, Y. Zhang, L. Wang, N. Crespi, G. M. Lee, and T. Wu, "When traffic flow prediction and wireless big data analytics meet," *IEEE Netw.*, to be published.
[11] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 1451–1455.
[12] 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.
[13] Z. Jiang and S. Mao, "Energy delay tradeoff in cloud offloading for multi-core mobile devices," *IEEE Access*, vol. 3, pp. 2306–2316, 2015.
[14] C.-F. Liu, M. Bennis, and H. V. Poor, "Latency and reliability-aware task offloading and resource allocation for mobile edge computing," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2017, pp. 1–7.
[15] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 3, pp. 361–373, Sep. 2017.
[16] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.
[17] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[18] 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. 2016.
[19] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Trans. Services Comput.*, to be published.
[20] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet Things J.*, to be published.
[21] N. Mastronarde and M. van der Schaar, "Joint physical-layer and system-level power management for delay-sensitive wireless communications," *IEEE Trans. Mobile Comput.*, vol. 12, no. 4, pp. 694–709, Apr. 2013.
[22] N. Salodkar, A. Bhorkar, A. Karandikar, and V. S. Borkar, "An online learning algorithm for energy efficient delay constrained scheduling over a fading channel," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 4, pp. 732–742, May 2008.
[23] F. Fu and M. V. D. Schaar, "Learning to compete for resources in wireless stochastic games," *IEEE Trans. Veh. Technol.*, vol. 58, no. 4, pp. 1904–1919, May 2009.
[24] V. S. Borkar and K. Soumyanatha, "An analog scheme for fixed point computation. I. Theory," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 44, no. 4, pp. 351–355, Apr. 1997.
[25] J. N. Tsitsiklis, "Asynchronous stochastic approximation and Q-learning," *Mach. Learn.*, vol. 16, no. 16, pp. 185–202, Sep. 1994.

XIAO ZHENG received the B.S. degree from Shandong University, in 2010, and the M.S. degree from Xihua University, Chengdu, China, in 2016. She is currently pursuing the Ph.D. degree with the School of Software, Dalian University of Technology. Her current research interests include mobile edge computing and Game Theory.

MINGCHU LI received the B.S. degree in mathematics fromo Jiangxi Normal University, in 1983, the M.S. degree in applied science from the University of Science and Technology Beijing, in 1989, and the Ph.D. degree in mathematics from the University of Toronto, in 1997. He was an Associate Professor with the University of Science and Technology Beijing, from 1989 to 1994. He was engaged in the research and development of information security at Long view Solution Inc. and Compute ware Inc., from 1997 to 2002. Since 2002, he has been with the School of Software, Tianjin University, as a Full Professor, and since 2004, he has been with the School of Software Technology, Dalian University of Technology, as a Full Professor, a Ph.D. Supervisor, and the Vice Dean. His main research interests include theoretical computer science and cryptography.

MUHAMMAD TAHIR received the B.S. degree in software engineering from the University of Sindh, Jamshoro Sindh, Pakistan, in 2008, and the M.S. degree in software engineering from the School of Software Engineering, Chongqing University, China, in 2014. He is currently pursuing the Ph.D. degree in software engineering with the School of Software Technology, Dalian University of Technology, China. He is on Ph.D. Study leave from Lecturer position with the Department of Computer Science, COMSATS University Islamabad, Sahiwal Campus, Pakistan. He has authored/coauthored publications in World renowned journals. His research interests include network security, web application performance tuning, mobile edge computing, game theory, artificial intelligence, and machine learning.

YUANFANG CHEN received the M.S. and Ph.D. degrees from the Dalian University of Technology, China, and the second Ph.D. degree from University Pierre and Marie CURIE (Paris VI), France. She is currently with Hangzhou Dianzi University, as a Professor. She was an Assistant Researcher of the Illinois Institute of Technology, USA, along with Prof. X. Li. She has been invited as the Session Chair of some conferences, the Associate Editor of Industrial Networks and Intelligent Systems, and the Guest Editor of the MONET.

MUHAMMAD ALAM received the Ph.D. degree in computer science from the University of Aveiro, Portugal, in 2014, with a specialization in Inter Layer and Cooperative Design Strategies for Green Mobile Networks. In 2009, he joined the Instituto de Telecomunicacoes-Aveiro, Portugal, as a Researcher. He has participated in several European Union FP7 projects, such as Hurricane, C2POWER, ICSI, and PEACE and Portuguese government funded projects such SmartVision. He is currently a Senior Researcher with the Instituto de Telecomunicacoes and participating in European Union and Portuguese government funded projects. His research interests include the IoT, real-time wireless communication, 5G, vehicular networks, context-aware systems, and radio resource management in next generation wireless networks. He is the Editor of the Book *Intelligent Transportation Systems, Dependable Vehicular Communications for Improved Road Safety*.