This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2876279, IEEE Internet of Things Journal

Optimized Computation Offloading Performance in Virtual Edge Computing Systems via Deep Reinforcement Learning

Xianfu Chen, *Member*, *IEEE*, Honggang Zhang, *Senior Member*, *IEEE*, Celimuge Wu, *Member*, *IEEE*, Shiwen Mao, *Senior Member*, *IEEE*, Yusheng Ji, *Senior Member*, *IEEE*, and Mehdi Bennis, *Senior Member*, *IEEE*

***Abstract*—To improve the quality of computation experience for mobile devices, mobile-edge computing (MEC) is a promising paradigm by providing computing capabilities in close proximity within a sliced radio access network (RAN), which supports both traditional communication and MEC services. Nevertheless, the design of computation offloading policies for a virtual MEC system remains challenging. Specifically, whether to execute a computation task at the mobile device or to offload it for MEC server execution should adapt to the time-varying network dynamics. This paper considers MEC for a representative mobile user in an ultra-dense sliced RAN, where multiple base stations (BSs) are available to be selected for computation offloading. The problem of solving an optimal computation offloading policy is modelled as a Markov decision process, where our objective is to maximize the long-term utility performance whereby an offloading decision is made based on the task queue state, the energy queue state as well as the channel qualities between MU and BSs. To break the curse of high dimensionality in state space, we first propose a double deep Q-network (DQN) based strategic computation offloading algorithm to learn the optimal policy without knowing a priori knowledge of network dynamics. Then motivated by the additive structure of the utility function, a Q-function decomposition technique is combined with the double DQN, which leads to a novel learning algorithm for the solving of stochastic computation offloading. Numerical experiments show that our proposed learning algorithms achieve a significant improvement in computation offloading performance compared with the baseline policies.**

***Index Terms*—Network slicing, radio access networks, network virtualization, mobile-edge computing, Markov decision process, deep reinforcement learning, Q-function decomposition.**

I. Introduction

With the proliferation of smart mobile devices, a multitude of mobile applications are emerging and gaining popularity, such as location-based virtual/augmented reality and online gaming [1]. However, mobile devices are in general resource-constrained, for example, the battery capacity and the local CPU computation power are limited. When executed at the mobile devices, the performance and Quality-of-Experience (QoE) of computation-intensive applications are significantly affected by the devices' limited computation capabilities. The tension between computation-intensive applications and resource-constrained mobile devices creates a bottleneck for having a satisfactory Quality-of-Service (QoS) and QoE, and is hence driving a revolution in computing infrastructure [2].

In contrast to cloud/fog computing [3], [4], mobile-edge computing (MEC) is envisioned as a promising paradigm, which provides computing capabilities within the radio access networks (RANs) in close proximity to mobile users (MUs) [5], [6]. By offloading computation tasks to the resource-rich MEC servers, not only the computation QoS and QoE can be greatly improved, but the capabilities of mobile devices can be augmented for running a variety of resource-demanding applications. Recently, lots of efforts have been put to the design of computation offloading policies. In [7], Wang et al. developed an alternating direction method of multipliers-based algorithm to solve the problem of revenue maximization by optimizing computation offloading decision, resource allocation and content caching strategy. In [8], Hu et al. proposed a two-phase based method for joint power and time allocation when considering cooperative computation offloading in a wireless power transfer-assisted MEC system. In [9], Wang et al. leveraged a Lagrangian duality method to minimize the total energy consumption in a computation latency constrained wireless powered multiuser MEC system.

For a MEC system, the computation offloading requires wireless data transmission, hence how to allocate wireless radio resource between the traditional communication service and the MEC service over a common RAN raises a series of technical challenges. Network slicing is a key enabler for RAN sharing, with which the traditional single ownership of network infrastructure and spectrum resources can be decoupled from the wireless services [10]. Consequently, the same physical network infrastructure is able to host multiple wireless service providers (WSPs) [11], [12]. In literature, there exist several efforts investigating joint communication and computation resource management in such virtualized networks, which support both the traditional communication service and the MEC service [13], [14]. In this work, we focus on designing optimal stochastic computation offloading policies in a sliced RAN, where a centralized network controller (CNC) is responsible for control-plane decisions on wireless radio resource orchestration over the traditional communication and MEC services.

The computation offloading policy designs in previous works [7], [8], [13]–[16] are mostly based on one-shot optimization and fail to characterize long-term computation offloading performance. In a virtual MEC system, the design of computation offloading policies should account for the environmental dynamics, such as the time-varying channel

X. Chen is with the VTT Technical Research Centre of Finland, Finland (e-mail: xianfu.chen@vtt.fi). H. Zhang is with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China (e-mail: honggangzhang@zju.edu.cn). C. Wu is with the Graduate School of Informatics and Engineering, University of Electro-Communications, Tokyo, Japan (email: clmg@is.uec.ac.jp). S. Mao is with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL, USA (email: smao@ieee.org). Y. Ji is with the Information Systems Architecture Research Division, National Institute of Informatics, Tokyo, Japan (e-mail: kei@nii.ac.jp). M. Bennis is with the Centre for Wireless Communications, University of Oulu, Finland (email: bennis@ee.oulu.fi).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2876279, IEEE Internet of Things Journal

quality and the task arrival and energy status at a mobile device. In [17], Liu et al. formulated the problem of delay-optimal computation task offloading under a Markov decision process (MDP) framework and developed an efficient one-dimensional search algorithm to find the optimal solution. However, the challenge lies in the dependence on statistical information of channel quality variations and computation task arrivals. In [18], Mao et al. investigated a dynamic computation offloading policy for a MEC system with wireless energy harvesting-enabled mobile devices using a Lyapunov optimization technique. The same technique was adopted to study the power-delay tradeoff in the scenario of computation task offloading by Liu et al. [19] and Jiang et al. [20]. The Lyapunov optimization can only construct an approximately optimal solution. Xu et al. developed in [21] a reinforcement learning based algorithm to learn the optimal computation offloading policy, which at the same time does not need a priori knowledge of network statistics.

When the MEC meets an ultra-dense sliced RAN, multiple base stations (BSs) with different data transmission qualities are available for offloading a computation task. In this context, the explosion in state space makes the conventional reinforcement learning algorithms [21]–[23] infeasible. Moreover, in this paper, wireless charging [24] is integrated into a MEC system, which on one hand achieves sustained computation performance but, on the other hand, makes the design of a stochastic computation offloading policy even more challenging. The main contributions in this work are fourfold.

- We formulate the stochastic computation offloading problem in a sliced RAN as a MDP, in which the time-varying communication qualities and computation resources are taken into account.
- To deal with the curse of state space explosion, we resort to a deep neural network based function approximator [25] and derive a double deep Q-network (DQN) [26] based reinforcement learning (DARLING) algorithm to learn the optimal computation offloading policy without any a priori knowledge of network dynamics.
- By further exploring the additive structure of the utility function, we attain a novel online deep state-action-reward-state-action based reinforcement learning algorithm (Deep-SARL) for the problem of stochastic computation offloading. To the best knowledge of the authors, this is the first work to combine a Q-function decomposition technique with the double DQN.
- Numerical experiments based on TensorFlow [27] are conducted to verify the theoretical studies in this paper. It shows that both of our proposed online learning algorithms outperform three baseline schemes. Especially, the Deep-SARL algorithm achieves the best computation offloading performance.

The rest of the paper is organized as follows. In the next section, we describe the system model and the assumptions made throughout this paper. In Section III, we formulate the problem of designing an optimal stochastic computation offloading policy as a MDP. We detail the derived online learning algorithms for stochastic computation offloading in a virtual MEC system in Section IV. To validate the proposed studies, we provide numerical experiments under various settings in Section V. Finally, we draw the conclusions in Section VI.

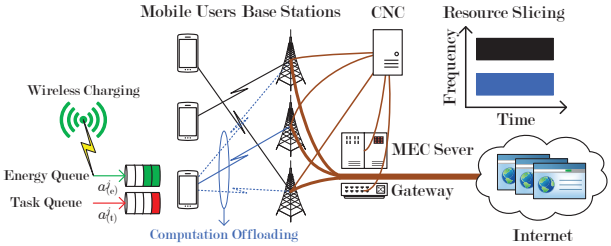


Fig. 1. Illustration of mobile-edge computing (MEC) in a virtualized radio access network, where the devices of mobile users are wireless charging enabled, the radio resource is sliced between conventional communication services (the links in black color) and MEC services (the links in blue color), and a centralized network controller (CNC) is responsible for all control plane decisions over the network.

II. System Descriptions and Assumptions

As illustrated in Fig. 1, we shall consider in this paper an ultra-dense service area covered by a virtualized RAN with a set $\mathcal{B} = \{1, \cdots, B\}$ of BSs. Both traditional communication services and MEC services are supported over the common physical network infrastructure. A MEC server is implemented at the network edge, providing rich computing resources for the MUs. By strategically offloading the generated computation tasks via the BSs to the MEC server for execution, the MUs can expect a significantly improved computation experience. We assume that the wireless radio resources are divided into traditional communication and MEC slices to guarantee inter-slice isolation. All control plane operations happening in such a hybrid network are managed by the CNC. The focus of this work is to optimize computation performance from a perspective of the MUs, while the design of joint traditional communication and MEC resource allocation is left for our next-step investigation. In a dense networking area, our analysis hereinafter concentrates on a representative MU. The time horizon is discretized into decision epochs, each of which is of equal duration δ (in seconds) and is indexed by an integer $j \in \mathbb{N}_+$. Let W (in Hz) denote the frequency bandwidth allocated to the MEC slice, which is shared among the MUs simultaneously accessing the MEC service.

This work assumes that the mobile device of the MU is wireless charging enabled and the received energy can be stored in an energy queue. The computation task generated by the MU across the time horizon form an independent and identically distributed sequence of Bernoulli random variables with a common parameter $\lambda_{(\mathrm{t})} \in [0, 1]$. We denote $a^j_{(\mathrm{t})} \in \{0, 1\}$ as the task arrival indicator, that is, $a^j_{(\mathrm{t})} = 1$ if a computation task is generated from the MU during a decision epoch j and otherwise $a^j_{(\mathrm{t})} = 0$. Then, $\mathsf{Pr}\{a^j_{(\mathrm{t})} = 1\} = 1 - \mathsf{Pr}\{a^j_{(\mathrm{t})} = 0\} = \lambda_{(\mathrm{t})}$, where $\mathsf{Pr}\{\cdot\}$ denotes the probability of the occurrence of an event. We represent a computation task by (μ, ν) with μ and ν being, respectively, the input data size (in bits) and the total number of CPU cycles required to accomplish the task. A computation task generated at a current decision epoch can be executed starting from the next epoch. The generated but

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2876279, IEEE Internet of Things Journal

not processed computation tasks can be queued at the mobile device of the MU. Based on a first-in first-out principle, a computation task from the task queue can be scheduled for execution either locally on the mobile device or remotely at the MEC server. More specifically, at the beginning of each decision epoch j, the MU makes a joint control action (c^j, e^j), where $c^j \in \{0\} \cup \mathcal{B}$ is the computation offloading decision and $e^j \in \mathbb{N}_+$ is the number of allocated energy units[1]. We have $c^j > 0$ if the MU chooses to offload the scheduled computation task to the MEC server via BS $c^j \in \mathcal{B}$ and $c^j = 0$ if the MU decides to execute the computation task locally on its own mobile device. Note that when $e^j = 0$, the queued tasks will not be executed.

When a computation task is scheduled for processing locally at the mobile device of the MU during a decision epoch j, i.e., $c^j = 0$, the allocated CPU-cycle frequency with $e^j > 0$ energy units can be calculated as

$$f^j = \sqrt{\frac{e^j}{\tau \cdot \nu}},$$

where τ is the effective switched capacitance that depends on chip architecture of the mobile device [28]. Moreover, the CPU-cycle frequency is constrained by $f^j \leq f^{(\max)}_{(\mathrm{CPU})}$. Then the time needed for local computation task execution is given by

$$d^j_{(\mathrm{mobile})} = \frac{\nu}{fj}, \tag{1}$$

which decreases as the number of allocated energy units increases.

We denote g^j_b as the channel gain state between the MU and a BS $b \in \mathcal{B}$ during each decision epoch j, which independently picks a value from a finite state space $\mathcal{G}_b$. The channel state transitions across the time horizon are modelled as a finite-state discrete-time Markov chain. At the beginning of a decision epoch j, if the MU lets the MEC server execute the scheduled computation task on behalf of the mobile device, the input data of the task needs to be offloaded to the chosen BS $c^j \in \mathcal{B}$. The MU-BS association has to be first established. If the chosen BS c^j is different from the previously associated one, a handover between the two BSs hence happens [29]. Denote $s^j \in \mathcal{B}$ as the MU-BS association state at a decision epoch j[2],

$$s^j = b \cdot \mathbf{1}_{\{\{c^{j-1}=b, b\in\mathcal{B}\} \vee \{\{c^{j-1}=0\} \wedge \{s^{j-1}=b\}\}\}},$$

where the symbols $\vee$ and $\wedge$ mean "logic OR" and "logic AND", respectively, and $\mathbf{1}_{\{\Omega\}}$ is the indicator function that equals 1 if the condition Ω is satisfied and otherwise 0. We assume that the energy consumption during the handover procedure is negligible at the mobile device. In our considered dense networking scenario, the achievable data rate can be written as

$$r^j = W \cdot \log_2\left(1 + I^{-1} \cdot g^j_b \cdot p^j_{(\mathrm{tr})}\right), \tag{2}$$

where I is the received average power of interference plus additive background Gaussian noise and

$$p^j_{(\mathrm{tr})} = \frac{e^j}{d^j_{(\mathrm{tr})}},$$

is the transmit power with

$$d^j_{(\mathrm{tr})} = \frac{\mu}{r^j}, \tag{3}$$

being the time of transmitting task input data. The transmit power is constrained by the maximum transmit power of the mobile device $p^{(\max)}_{(\mathrm{tr})}$ [30], i.e.,

$$p^j_{(\mathrm{tr})} \leq p^{(\max)}_{(\mathrm{tr})}. \tag{4}$$

In (3) above, we assume that the energy is evenly assigned to the input data bits of the computation task. In other words, the transmission rate keeps unchanged during the input data transmission. Lemma 1 ensures that $d^j_{(\mathrm{tr})}$ is the minimum transmission time given the allocated energy units $e^j > 0$.

Lemma 1: Given the computation offloading decision $c^j \in \mathcal{B}$ and the allocated energy units $e^j > 0$ at a decision epoch j, the optimal transmission policy achieving the minimum transmission time is a policy with which the rate of transmitting task input data remains a constant.

Proof: The proof is given as Appendix A. □

Lemma 2: Given the computation offloading decision $c^j \in \mathcal{B}$ at a decision epoch j, the input data transmission time $d^j_{(\mathrm{tr})}$ is a monotonically decreasing function of the allocated energy units $e^j > 0$.

Proof: The proof is given in Appendix B. □

In addition, we assume in this paper that the battery capacity at the mobile device of the MU is limited and the received energy units across the time horizon take integer values. Let $q^j_{(\mathrm{e})}$ be the energy queue length of the MU at the beginning of a decision epoch j, which evolves according to

$$q^{j+1}_{(\mathrm{e})} = \min\left\{q^j_{(\mathrm{e})} - e^j + a^j_{(\mathrm{e})}, q^{(\max)}_{(\mathrm{e})}\right\},$$

where $q^{(\max)}_{(\mathrm{e})} \in \mathbb{N}_+$ denotes the battery capacity limit and $a^j_{(\mathrm{e})} \in \mathbb{N}_+$ is the number of energy units received by the end of decision epoch j.

III. Problem Formulation

In this section, we shall first formulate the problem of stochastic computation offloading within the MDP framework and then discuss the optimal solutions.

A. Stochastic Computation Task Offloading

The experienced delay is the key performance indicator for evaluating the quality of a task computing experience. The delay of a computation task is defined as the period of time from when the task arrives to the computation task queue to when the task is successfully removed from the task queue. Thus the experienced delay includes the computation task execution delay and the task queuing delay. We assume that there is a delay of ζ seconds for control signalling during the occurrence of one handover. With a joint control action (c^j, e^j)

[1]An energy unit corresponds to an amount of energy, say, $2 \cdot 10^{-3}$ Joules as in numerical experiments.

[2]We assume that if the MU processes a computation task locally or no task is executed at a decision epoch $j-1$, then the MU-BS association does not change, namely, $s^j = s^{j-1}$. In this case, no handover will be triggered.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2876279, IEEE Internet of Things Journal

at a decision epoch j, the handover delay can be then given as

$$h^j = \zeta \cdot \mathbf{1}_{\{\{c^j \in \mathcal{B}\} \wedge \{c^j \neq s^j\}\}}. \tag{5}$$

According to (1), (3) and (5), we obtain the task execution delay as[3]

$$d^j = \begin{cases} d^j_{(\text{mobile})}, & \text{if } e^j > 0 \text{ and } c^j = 0; \\ h^j + d^j_{(\text{tr})} + d_{(\text{server})}, & \text{if } e^j > 0 \text{ and } c^j \in \mathcal{B}; \\ 0, & \text{if } e^j = 0, \end{cases}$$

where $d_{(\text{server})}$ is time consumed for task execution at the MEC server. Due to the sufficient available computation resource at the MEC server, we assume that $d_{(\text{server})}$ is a sufficiently small constant.

Notice that if: 1) the MU fails to process a computation task at the mobile device within one decision epoch; or 2) a computation task is scheduled for MEC server execution but the computation result cannot be sent back via the chosen BS within the decision epoch, the task execution fails and the task will remain in the queue until being successfully executed. The dynamics of the computation task queue at the MU can be hence expressed as

$$q^{j+1}_{(\text{t})} = \min\Big\{ q^j_{(\text{t})} - \mathbf{1}_{\{0 < d^j \leq \delta\}} + a^j_{(\text{t})}, q^{(\max)}_{(\text{t})} \Big\},$$

where $q^j_{(\text{t})}$ is the number of computation tasks in the queue at the beginning of each decision epoch j and $q^{(\max)}_{(\text{t})} \in \mathbb{N}_+$ limits the maximum number of computation tasks that can be queued at the mobile device. There will be computation task drops once the task queue is full. We let

$$\eta^j = \max\Big\{ q^j_{(\text{t})} - \mathbf{1}_{\{0 < d^j \leq \delta\}} + a^j_{(\text{t})} - q^{(\max)}_{(\text{t})}, 0 \Big\},$$

define a computation task drop.

If a computation task remains in the queue for a decision epoch, a delay of δ seconds will be incurred to the task. We treat the queuing delay during a decision epoch j equivalently as the length of a task queue, that is,

$$\rho^j = q^j_{(\text{t})} - \mathbf{1}_{\{d^j > 0\}}.$$

As previously discussed, if $d^j > \delta$, the execution of a computation task fails. In this case, the MU receives a penalty, which is defined by

$$\varphi^j = \mathbf{1}_{\{d^j > \delta\}}.$$

Moreover, a payment is required for the access to MEC service when the MU decides to offload a computation task for MEC server execution. The payment is assumed to be proportional to the time consumed for transmitting and processing the task input data. That is, the payment can be calculated as

$$\phi^j = \pi \cdot \big(\min\{d^j, \delta\} - h^j \big) \cdot \mathbf{1}_{\{c^j \in \mathcal{B}\}},$$

where $\pi \in \mathbb{R}_+$ is the price paid for the MEC service per unit of time.

The network state of the MU during each decision epoch j can be characterized by $\boldsymbol{\chi}^j = (q^j_{(\text{t})}, q^j_{(\text{e})}, s^j, \mathbf{g}^j) \in \mathcal{X} \stackrel{\text{def}}{=} \{0, 1, \cdots, q^{(\max)}_{(\text{t})}\} \times \{0, 1, \cdots, q^{(\max)}_{(\text{e})}\} \times \mathcal{B} \times \{\times_{b \in \mathcal{B}} \mathcal{G}_b\}$, where $\mathbf{g}^j = (g^j_b : b \in \mathcal{B})$. At the beginning of epoch j, the MU decides a joint task offloading and energy allocation decision $(c^j, e^j) \in \mathcal{Y} \stackrel{\text{def}}{=} \{\{0\} \cup \mathcal{B}\} \times \{0, 1, \cdots, Q_{(\text{e})}\}$[4] according to the stationary control policy defined by Definition 1. In line with the discussions, we define an immediate utility at epoch j to quantify the task computation experience for the MU,

$$\begin{aligned} u\big(\boldsymbol{\chi}^j, (c^j, e^j)\big) = {} & \omega_1 \cdot u^{(1)}\big(\min\{d^j, \delta\}\big) + \omega_2 \cdot u^{(2)}(\eta^j) \\ & + \omega_3 \cdot u^{(3)}(\rho^j) + \omega_4 \cdot u^{(4)}(\varphi^j) \\ & + \omega_5 \cdot u^{(5)}(\phi^j), \end{aligned} \tag{6}$$

where the positive monotonically deceasing functions $u^{(1)}(\cdot)$, $u^{(2)}(\cdot)$, $u^{(3)}(\cdot)$, $u^{(4)}(\cdot)$ and $u^{(5)}(\cdot)$ measure the satisfactions of the task execution delay, the computation task drops, the task queuing delay, the penalty of failing to execute a computation task and the payment of accessing the MEC service, and ω_1, ω_2, ω_3, ω_4, $\omega_5 \in \mathbb{R}_+$ are the weights that combine different types of function with different units into a universal utility function. With slight abuse of notations, we rewrite $u^{(1)}(\cdot)$, $u^{(2)}(\cdot)$, $u^{(3)}(\cdot)$, $u^{(4)}(\cdot)$ and $u^{(5)}(\cdot)$ as $u^{(1)}(\boldsymbol{\chi}^j, (c^j, e^j))$, $u^{(2)}(\boldsymbol{\chi}^j, (c^j, e^j))$, $u^{(3)}(\boldsymbol{\chi}^j, (c^j, e^j))$, $u^{(4)}(\boldsymbol{\chi}^j, (c^j, e^j))$ and $u^{(5)}(\boldsymbol{\chi}^j, (c^j, e^j))$.

Definition 1 (Joint Task Offloading and Energy Allocation Control Policy): A stationary joint task offloading and energy allocation control policy $\boldsymbol{\Phi}$ is defined as a mapping: $\boldsymbol{\Phi} : \mathcal{X} \to \mathcal{Y}$. More specifically, the MU determines a joint control action $\boldsymbol{\Phi}(\boldsymbol{\chi}^j) = (\Phi_{(\text{c})}(\boldsymbol{\chi}^j), \Phi_{(\text{e})}(\boldsymbol{\chi}^j)) = (c^j, e^j) \in \mathcal{Y}$ according to $\boldsymbol{\Phi}$ after observing network state $\boldsymbol{\chi}^j \in \mathcal{X}$ at the beginning of each decision epoch j, where $\boldsymbol{\Phi} = (\Phi_{(\text{c})}, \Phi_{(\text{e})})$ with $\Phi_{(\text{c})}$ and $\Phi_{(\text{e})}$ being, respectively, the stationary task offloading and energy allocation policies.

Given a stationary control policy $\boldsymbol{\Phi}$, the $\{\boldsymbol{\chi}^j : j \in \mathbb{N}_+\}$ is a controlled Markov chain with the following state transition probability

$$\begin{aligned} & \mathsf{Pr}\{\boldsymbol{\chi}^{j+1} | \boldsymbol{\chi}^j, \boldsymbol{\Phi}(\boldsymbol{\chi}^j)\} = \\ & \mathsf{Pr}\Big\{ q^{j+1}_{(\text{t})} | q^j_{(\text{t})}, \boldsymbol{\Phi}(\boldsymbol{\chi}^j) \Big\} \cdot \mathsf{Pr}\Big\{ q^{j+1}_{(\text{e})} | q^j_{(\text{e})}, \boldsymbol{\Phi}(\boldsymbol{\chi}^j) \Big\} \cdot \\ & \mathsf{Pr}\big\{ s^{j+1} | s^j, \boldsymbol{\Phi}(\boldsymbol{\chi}^j) \big\} \cdot \prod_{b \in \mathcal{B}} \mathsf{Pr}\Big\{ g^{j+1}_b | g^j_b \Big\}. \end{aligned}$$

Taking expectation with respect to the per-epoch utilities $\{u(\boldsymbol{\chi}^j, \boldsymbol{\Phi}(\boldsymbol{\chi}^j)) : j \in \mathbb{N}_+\}$ over the sequence of network states $\{\boldsymbol{\chi}^j : j \in \mathbb{N}_+\}$, the expected long-term utility of the MU conditioned on an initial network state $\boldsymbol{\chi}^1$ can be expressed as

$$\begin{aligned} & V(\boldsymbol{\chi}, \boldsymbol{\Phi}) = \\ & \mathsf{E}_{\boldsymbol{\Phi}} \left[(1 - \gamma) \cdot \sum_{j=1}^{\infty} (\gamma)^{j-1} \cdot u\big(\boldsymbol{\chi}^j, \boldsymbol{\Phi}\left(\boldsymbol{\chi}^j\right)\big) \, | \boldsymbol{\chi}^1 = \boldsymbol{\chi} \right], \end{aligned} \tag{7}$$

[3]In this work, we assume that the BSs are connected to the MEC server via fibre links. Hence, the round-trip delay between the BSs and the MEC server is negligible. Further, we neglect the time overhead for the selected BS to send back the computation result due to the fact that the size of a computation outcome is much smaller than the input data of a computation task [31].

[4]To keep what follows uniform, we do not exclude the infeasible joint actions.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2876279, IEEE Internet of Things Journal

where $\boldsymbol{\chi} = (q_{(\mathrm{t})}, q_{(\mathrm{e})}, s, \mathbf{g}) \in \mathcal{X}$, $\mathbf{g} = (g_b : b \in \mathcal{B})$, $\gamma \in [0, 1)$ is the discount factor, and $(\gamma)^{j-1}$ denotes the discount factor to the $(j-1)$-th power. $V(\boldsymbol{\chi}, \boldsymbol{\Phi})$ is also named as the state-value function for the MU in the state $\boldsymbol{\chi}$ under policy $\boldsymbol{\Phi}$.

Problem 1: The MU aims to design an optimal stationary control policy $\boldsymbol{\Phi}^* = (\Phi^*_{(\mathrm{c})}, \Phi^*_{(\mathrm{e})})$ that maximizes the expected long-term utility performance, $V(\boldsymbol{\chi}, \boldsymbol{\Phi})$, for any given initial network state $\boldsymbol{\chi}$, which can be formally formulated as in the following

$$\boldsymbol{\Phi}^* = \arg\max_{\boldsymbol{\Phi}} V(\boldsymbol{\chi}, \boldsymbol{\Phi}), \forall \boldsymbol{\chi} \in \mathcal{X}.$$

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

Remark 1: The formulated problem of stochastic computation offloading optimization as in Problem 1 is in general a single-agent infinite-horizon MDP with the discounted utility criterion. Nevertheless, (7) can also be used to approximate the expected infinite-horizon undiscounted utility [32]

$$U(\boldsymbol{\chi}, \boldsymbol{\Phi}) = \mathsf{E}_{\boldsymbol{\Phi}}\left[\lim_{J\to\infty} \frac{1}{J} \cdot \sum_{j=1}^{J} u(\boldsymbol{\chi}^j, \boldsymbol{\Phi}(\boldsymbol{\chi}^j)) | \boldsymbol{\chi}^1 = \boldsymbol{\chi}\right].$$

when γ approaches 1.

B. Learning Optimal Solution to Problem 1

The stationary control policy achieving the optimal state-value function can be obtained by solving the following Bellman's optimality equation [23]: $\forall \boldsymbol{\chi} \in \mathcal{X}$,

$$V(\boldsymbol{\chi}) = \max_{(c,e)} \Big\{ (1-\gamma) \cdot u(\boldsymbol{\chi}, (c, e)) + \gamma \cdot \sum_{\boldsymbol{\chi}'} \mathsf{Pr}\{\boldsymbol{\chi}' | \boldsymbol{\chi}, (c, e)\} \cdot V(\boldsymbol{\chi}') \Big\}, \tag{8}$$

where $u(\boldsymbol{\chi}, (c, e))$ is the achieved utility when a joint control action $(c, e) \in \mathcal{Y}$ is performed under network state $\boldsymbol{\chi}$ and $\boldsymbol{\chi}' = (q'_{(\mathrm{t})}, q'_{(\mathrm{e})}, s', \mathbf{g}') \in \mathcal{X}$ is the subsequent network state with $\mathbf{g}' = (g'_b : b \in \mathcal{B})$.

Remark 2: The traditional solutions to (8) are based on the value iteration or the policy iteration [23], which need *complete knowledge* of the computation task arrival, the received energy unit and the channel state transition statistics.

One attractiveness of the off-policy Q-learning is that it assumes no a priori knowledge of the network state transition statistics [23]. We define the right-hand side of (8) by

$$Q(\boldsymbol{\chi}, (c, e)) = (1-\gamma) \cdot u(\boldsymbol{\chi}, (c, e)) + \gamma \cdot \sum_{\boldsymbol{\chi}'} \mathsf{Pr}\{\boldsymbol{\chi}' | \boldsymbol{\chi}, (c, e)\} \cdot V(\boldsymbol{\chi}'), \tag{9}$$

$\forall \boldsymbol{\chi} \in \mathcal{X}$. The optimal state-value function $V(\boldsymbol{\chi})$ can be hence directly obtained from

$$V(\boldsymbol{\chi}) = \max_{(c,e)} Q(\boldsymbol{\chi}, (c, e)). \tag{10}$$

By substituting (10) into (9), we get

$$Q(\boldsymbol{\chi}, (c, e)) = (1-\gamma) \cdot u(\boldsymbol{\chi}, (c, e)) + \gamma \cdot \sum_{\boldsymbol{\chi}'} \mathsf{Pr}\{\boldsymbol{\chi}' | \boldsymbol{\chi}, (c, e)\} \cdot \max_{(c',e')} Q(\boldsymbol{\chi}', (c', e')),$$

where we denote $(c', e') \in \mathcal{Y}$ as a joint control action performed under the network state $\boldsymbol{\chi}'$. In practice, the computation task arrival and the number of energy units that can be received by the end of a decision epoch are unavailable beforehand. Using standard Q-learning, the MU tries to learn $Q(\boldsymbol{\chi}, (c, e))$ in a recursive way based on the observation of network state $\boldsymbol{\chi} = \boldsymbol{\chi}^j$ at a current decision epoch j, the performed joint action $(c, e) = (c^j, e^j)$, the achieved utility $u(\boldsymbol{\chi}, (c, e))$ and the resulting network state $\boldsymbol{\chi}' = \boldsymbol{\chi}^{j+1}$ at the next epoch $j+1$. The updating rule is given in (11), where $\alpha^j \in [0, 1)$ is a time-varying learning rate. It has been proven that if 1) the network state transition probability under the optimal stationary control policy is stationary, 2) $\sum_{j=1}^{\infty} \alpha^j$ is infinite and $\sum_{j=1}^{\infty} (\alpha^j)^2$ is finite, and 3) all state-action pairs are visited infinitely often, the Q-learning process converges and eventually finds the optimal control policy [22]. The last condition can be satisfied if the probability of choosing any action in any network state is non-zero (i.e., *exploration*). Meanwhile, the MU has to exploit the most recent knowledge of Q-function in order to perform well (i.e., *exploitation*). A classical way to balance the trade-off between *exploration* and *exploitation* is the ϵ-greedy strategy [23].

Remark 3: From (11), we can find that the standard Q-learning rule suffers from *poor scalability*. Due to the tabular nature in representing Q-function values, Q-learning is not readily applicable to high-dimensional scenarios with extremely huge network state and/or action spaces, where the learning process is extremely slow [33], [34]. In our considered system model, the sizes of the network state space $\mathcal{X}$ and the action space $\mathcal{Y}$ can be calculated as $X = (1+q^{(\max)}_{(\mathrm{t})}) \cdot (1+q^{(\max)}_{(\mathrm{e})}) \cdot B \cdot \prod_{b\in\mathcal{B}} |\mathcal{G}_b|$ and $Y = (1+B) \cdot (1+q^{(\max)}_{(\mathrm{e})})$, respectively, where $|\mathcal{G}|$ means the cardinality of the set $\mathcal{G}$. It can be observed that X grows exponentially as the number B of BSs increases. Suppose there is a MEC system with 6 BSs and for each BS, the channel gain is quantized into 6 states (as assumed in our experiment setups). If we set $q^{(\max)}_{(\mathrm{t})} = q^{(\max)}_{(\mathrm{e})} = 4$, the MU has to update in total $X \cdot Y = 2.44944 \cdot 10^8$ Q-function values during the learning process, which is impossible for the Q-learning process to converge within limited number of decision epoches.

The next section thereby focuses on developing practically feasible and computationally efficient algorithms to approach the optimal control policy.

IV. Approaching the Optimal Policy

In this section, we proceed to approach the optimal control policy by developing practically feasible algorithms based on recent advances in deep reinforcement learning and a linear Q-function decomposition technique.

A. Deep Reinforcement Learning Algorithm

Inspired by the success of modelling an optimal state-action Q-function with a deep neural network [25], we adopt a double DQN to address the massive network state space $\mathcal{X}$ [26]. Specifically, the Q-function expressed as in (9) is approximated by $Q(\boldsymbol{\chi}, (c, e)) \approx Q(\boldsymbol{\chi}, (c, e); \boldsymbol{\theta})$, where $(\boldsymbol{\chi}, (c, e)) \in$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2876279, IEEE Internet of Things Journal

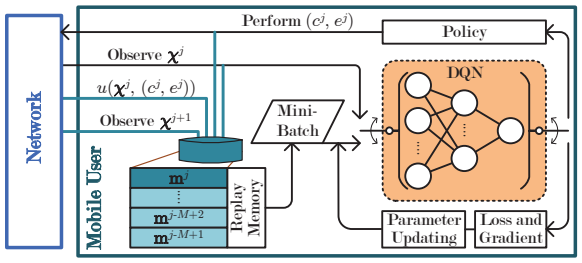


Fig. 2. Double deep Q-network (DQN) based reinforcement learning (DARLING) for stochastic computation offloading in a mobile-edge computing system.

$\mathcal{X}\times\mathcal{Y}$ and $\boldsymbol{\theta}$ denotes a vector of parameters associated with the DQN. The DQN-based reinforcement learning (DARLING) for stochastic computation offloading in our considered MEC system is illustrated in Fig. 2, during which instead of finding the optimal Q-function, the DQN parameters can be learned iteratively.

The mobile device is assumed to be equipped with a replay memory of a finite size M to store the experience $\mathbf{m}^j = (\boldsymbol{\chi}^j, (c^j, e^j), u(\boldsymbol{\chi}^j, (c^j, e^j)), \boldsymbol{\chi}^{j+1})$ that is happened at the transition of two consecutive decision epoches j and $j+1$ during the learning process of DARLING, where $\boldsymbol{\chi}^j$, $\boldsymbol{\chi}^{j+1} \in \mathcal{X}$ and $(c^j, e^j) \in \mathcal{Y}$. The experience pool can be represented as $\mathcal{M}^j = \{\mathbf{m}^{j-M+1}, \cdots, \mathbf{m}^j\}$. The MU maintains a DQN and a target DQN, namely, $Q(\boldsymbol{\chi}, (c, e); \boldsymbol{\theta}^j)$ and $Q(\boldsymbol{\chi}, (c, e); \boldsymbol{\theta}^j_-)$, with parameters $\boldsymbol{\theta}^j$ at a current decision epoch j and $\boldsymbol{\theta}^j_-$ at some previous epoch before decision epoch j, $\forall(\boldsymbol{\chi}, (c, e)) \in \mathcal{X} \times \mathcal{Y}$. According to the experience replay technique [36], the MU then randomly samples a mini-batch $\widetilde{\mathcal{M}}^j \subseteq \mathcal{M}^j$ from the pool $\mathcal{M}^j$ of historical experiences at each decision epoch j to online train the DQN. That is, the parameters $\boldsymbol{\theta}^j$ are updated in the direction of minimizing the loss function, which is defined by (12), where $(c', e') \in \mathcal{Y}$. The loss function $L_{(\text{DARLING})}(\boldsymbol{\theta}^j)$ is a mean-squared measure of the Bellman equation error at a decision epoch j (i.e., the last term of (11)) by replacing $Q^j(\boldsymbol{\chi}, (c, e))$ and its corresponding target $(1-\gamma)\cdot u(\boldsymbol{\chi}, (c, e)) + \gamma \cdot \max_{(c', e')} Q^j(\boldsymbol{\chi}', (c', e'))$ with $Q(\boldsymbol{\chi}, (c, e); \boldsymbol{\theta}^j)$ and $(1-\gamma)\cdot u(\boldsymbol{\chi}, (c, e)) + \gamma \cdot Q(\boldsymbol{\chi}', \arg\max_{(c', e')} Q(\boldsymbol{\chi}', (c', e'); \boldsymbol{\theta}^j); \boldsymbol{\theta}^j_-)$ [26], respectively. By differentiating the loss function $L_{(\text{DARLING})}(\boldsymbol{\theta}^j)$ with respect to the DQN parameters $\boldsymbol{\theta}^j$, we obtain the gradient as in (13). Algorithm 1 summarizes the implementation of the online DARLING algorithm by the MU for stochastic computation offloading in our considered MEC system.

Algorithm 1 Online DARLING Algorithm for Stochastic Computation Task Offloading in A MEC System

1: **input** A DQN and a target DQN with two sets $\boldsymbol{\theta}^j$ and $\boldsymbol{\theta}^j_-$ of random parameters, for $j = 1$.
2: **initialize** the replay memory $\mathcal{M}^j$ with a finite size of $M \in \mathbb{N}_+$, the mini-batch $\widetilde{\mathcal{M}}^j$ with a size of $\tilde{M} < M$ for experience replay, for $j = 1$.
3: **repeat**
4: At the beginning of decision epoch j, the MU observes the network state $\boldsymbol{\chi}^j \in \mathcal{X}$, which is taken as an input to the DQN with parameters $\boldsymbol{\theta}^j$, and then selects a joint control action $(c^j, e^j) \in \mathcal{Y}$ randomly with probability ϵ or $(c^j, e^j) = \arg\max_{(c,e)\in\mathcal{Y}} Q(\boldsymbol{\chi}^j, (c, e); \boldsymbol{\theta}^j)$ with probability $1-\epsilon$.
5: After performing the selected joint control action (c^j, e^j), the MU realizes an immediate utility $u(\boldsymbol{\chi}^j, (c^j, e^j))$ and observes the new network state $\boldsymbol{\chi}^{j+1} \in \mathcal{X}$ at the next decision epoch $j+1$.
6: The MU updates the replay memory $\mathcal{M}^j$ at the mobile device with the most recent transition $\mathbf{m}^j = (\boldsymbol{\chi}^j, (c^j, e^j), u(\boldsymbol{\chi}^j, (c^j, e^j)), \boldsymbol{\chi}^{j+1})$.
7: With a randomly sampled mini-batch of transitions $\widetilde{\mathcal{M}}^j \subseteq \mathcal{M}^j$ from the replay memory, the MU updates the DQN parameters $\boldsymbol{\theta}^j$ with the gradient given by (13).
8: The MU regularly resets the target DQN parameters with $\boldsymbol{\theta}^{j+1}_- = \boldsymbol{\theta}^j$, and otherwise $\boldsymbol{\theta}^{j+1}_- = \boldsymbol{\theta}^j_-$.
9: The decision epoch index is updated by $j \leftarrow j+1$.
10: **until** A predefined stopping condition is satisfied.
11: **return** The parameters $\boldsymbol{\theta}$ associated with the DQN of the MU.

B. Linear Q-Function Decomposition based Deep Reinforcement Learning

1) Linear Q-Function Decomposition: It can be found that the utility function in (6) is of an additive structure, which motivates us to linearly decompose the state-action Q-function, namely, $Q(\boldsymbol{\chi}, (c, e))$, $\forall(\boldsymbol{\chi}, (c, e)) \in \mathcal{X} \times \mathcal{Y}$, based on the pattern $\mathcal{K} = \{1, \cdots, K\}$ of classifying the satisfactions regarding the task execution delay, the computation task drops, the task queuing delay, the penalty of failing to process a computation task and the payment of using the MEC service. For example, we can divide the utility into four satisfaction categories, namely, $u(\boldsymbol{\chi}, (c, e)) = u_1(\boldsymbol{\chi}, (c, e)) + u_2(\boldsymbol{\chi}, (c, e)) + u_3(\boldsymbol{\chi}, (c, e)) + u_4(\boldsymbol{\chi}, (c, e))$ with $u_1(\boldsymbol{\chi}, (c, e)) = \omega_1 \cdot u^{(1)}(\boldsymbol{\chi}, (c, e)) + \omega_3 \cdot u^{(3)}(\boldsymbol{\chi}, (c, e))$, $u_2(\boldsymbol{\chi}, (c, e)) = \omega_2 \cdot u^{(2)}(\boldsymbol{\chi}, (c, e))$, $u_3(\boldsymbol{\chi}, (c, e)) = \omega_4 \cdot u^{(4)}(\boldsymbol{\chi}, (c, e))$ and $u_4(\boldsymbol{\chi}, (c, e)) = \omega_5 \cdot u^{(5)}(\boldsymbol{\chi}, (c, e))$, then $\mathcal{K} = \{1, 2, 3, 4\}$ forms a classification pattern. In our considered stochastic computation offloading scenario, it's easy to see that $K \leq 5$. Mathematically, $Q(\boldsymbol{\chi}, (c, e))$ is decomposed into [35]

$$Q(\boldsymbol{\chi}, (c, e)) = \sum_{k\in\mathcal{K}} Q_k(\boldsymbol{\chi}, (c, e)), \tag{14}$$

where the MU deploys a "virtual" agent $k \in \mathcal{K}$ to learn the optimal per-agent state-action Q-function $Q_k(\boldsymbol{\chi}, (c, e))$ that satisfies

$$\begin{aligned} Q_k(\boldsymbol{\chi}, (c, e)) = {} & (1-\gamma)\cdot u_k(\boldsymbol{\chi}, (c, e)) \\ & + \gamma \cdot \sum_{\boldsymbol{\chi}'} \mathsf{Pr}\{\boldsymbol{\chi}' | \boldsymbol{\chi}, (c, e)\} \cdot Q_k(\boldsymbol{\chi}', \boldsymbol{\Phi}^*(\boldsymbol{\chi}')), \end{aligned} \tag{15}$$

$$Q^{j+1}(\boldsymbol{\chi}, (c, e)) = Q^j(\boldsymbol{\chi}, (c, e)) + \alpha^j \left((1-\gamma)\cdot u(\boldsymbol{\chi}, (c, e)) + \gamma \cdot \max_{(c', e')} Q^j(\boldsymbol{\chi}', (c', e')) - Q^j(\boldsymbol{\chi}, (c, e)) \right) \tag{11}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2876279, IEEE Internet of Things Journal

with $u_k(\boldsymbol{\chi},(c,e))$ being the immediate utility related to a satisfaction category k. We emphasize that the optimal joint control action in (15) of an agent k across the time horizon should reflect the optimal control policy implemented by the MU. In other words, the MU makes an optimal joint control action decision $\boldsymbol{\Phi}^*(\boldsymbol{\chi})$ under a network state $\boldsymbol{\chi}$

$$\boldsymbol{\Phi}^*(\boldsymbol{\chi}) = \arg\max_{(c,e)} \sum_{k\in\mathcal{K}} Q_k(\boldsymbol{\chi},(c,e)),$$

to maximize the aggregated Q-function values from all the agents. We will see in Theorem 1 that the linear Q-function decomposition technique achieves the optimal solution to Problem 1.

Theorem 1: The linear Q-function decomposition approach as in (14) asserts the optimal computation offloading performance.

Proof: The proof is given in Appendix C. □

Remark 4: An apparent advantage of the linear Q-function decomposition technique is that it potentially simplifies the problem solving. Back to the example above, agent 2 learns the optimal expected long-term satisfaction measuring the computation task drops across the time horizon. It's obvious that a computation task drop η^j at a decision epoch j depends only on the task queue state $q^j_{(\mathrm{t})}$ (rather than the network state $\boldsymbol{\chi}^j \in \mathcal{X}$) and the performed joint control action (c^j, e^j) by the MU.

Recall that the joint control action of each agent should be in accordance with the optimal control policy of the MU, the Q-learning rule, which involves off-policy updates, is obviously not applicable to finding the optimal per-agent state-action Q-functions. The state-action-reward-state-action (SARSA) algorithm [23], [37], which applies on-policy updates, fosters a promising alternative, namely, a SARSA-based reinforcement learning (SARL). Having the observations of the network state $\boldsymbol{\chi} = \boldsymbol{\chi}^j$, the performed joint control action $(c,e) = (c^j, e^j)$ by the MU, the realized per-agent utilities $(u_k(\boldsymbol{\chi},(c,e)) : k \in \mathcal{K})$ at a current decision epoch j and the resulting network state $\boldsymbol{\chi}' = \boldsymbol{\chi}^{j+1}$, the joint control action $(c',e') = (c^{j+1}, e^{j+1})$ selected by the MU at the next epoch $j+1$, each agent $k \in \mathcal{K}$ updates the state-action Q-function on the fly according to (16), where different from off-policy Q-learning, (c',e') is an actually performed joint control action in the subsequent network state, rather than the hypothetical one that maximizes the per-agent state-action Q-function. Theorem 2 ensures that the SARL algorithm converges.

Theorem 2: The sequence $\{(Q^j_k(\boldsymbol{\chi},(c,e)) : \boldsymbol{\chi} \in \mathcal{X}, (c,e) \in \mathcal{Y} \text{ and } k \in \mathcal{K}) : j \in \mathbb{N}_+\}$ by SARL converges to the optimal per-agent state-action Q-functions $Q_k(\boldsymbol{\chi},(c,e))$, $\forall \boldsymbol{\chi} \in \mathcal{X}$, $\forall (c,e) \in \mathcal{Y}$ and $\forall k \in \mathcal{K}$ if and only if the state-action pairs $(\boldsymbol{\chi},(c,e)) \in \mathcal{X}\times\mathcal{Y}$ are visited infinitely often and the learning rate α^j satisfies: $\sum_{j=1}^{\infty} \alpha^j = \infty$ and $\sum_{j=1}^{\infty} (\alpha^j)^2 < \infty$.

Proof: The proof is given in Appendix D. □

Remark 5: Implementing the derived SARL algorithm, the size of Q-function faced by each agent remains the same as the standard Q-learning algorithm. From Remark 4, the linear Q-function decomposition technique provides the possibility of simplifying the solving of the stochastic computation offloading problem through introducing multiple "virtual" agents. Each agent learns the respective expected long-term satisfaction by exploiting the key network state information and is hence enabled to use a simpler DQN to approximate the optimal state-action Q-function.

2) Deep SARSA Reinforcement Learning (Deep-SARL): Applying the linear Q-function decomposition technique, the DQN $Q(\boldsymbol{\chi},(c,e);\boldsymbol{\theta})$, which approximates the optimal state-action Q-function $Q(\boldsymbol{\chi},(c,e))$, $\forall(\boldsymbol{\chi},(c,e)) \in \mathcal{X}\times\mathcal{Y}$, can be reexpressed as,

$$Q(\boldsymbol{\chi},(c,e);\boldsymbol{\theta}) = \sum_{k\in\mathcal{K}} Q_k(\boldsymbol{\chi},(c,e);\boldsymbol{\theta}_k),$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_k : k \in \mathcal{K})$ is a collection of parameters associated with the DQNs of all agents and $Q_k(\boldsymbol{\chi},(c,e);\boldsymbol{\theta}_k)$ (for $k \in \mathcal{K}$) is the per-agent DQN. Accordingly, we derive a novel deep SARSA reinforcement learning (Deep-SARL) based stochastic computation offloading algorithm, as depicted in Fig. 3, where different from the DARLING algorithm, the parameters $\boldsymbol{\theta}$ are learned locally at the agents in an online iterative way.

Implementing our proposed Deep-SARL algorithm, at each decision epoch j, the MU updates the experience pool $\mathcal{N}^j$ with the most recent N experiences $\{\mathbf{n}^{j-N+1},\cdots,\mathbf{n}^j\}$ with each experience $\mathbf{n}^j = (\boldsymbol{\chi}^j,(c^j,e^j),(u_k(\boldsymbol{\chi}^j,(c^j,e^j)) : k \in \mathcal{K}),\boldsymbol{\chi}^{j+1},(c^{j+1},e^{j+1}))$. To train the DQN parameters, the MU first randomly samples a mini-batch $\widetilde{\mathcal{N}}^j \subseteq \mathcal{N}^j$ and then updates $\boldsymbol{\theta}^j = (\boldsymbol{\theta}^j_k : k \in \mathcal{K})$ for all agents at each decision epoch j to minimize the accumulative loss function, which is given by (17), where $\boldsymbol{\theta}^j_- = (\boldsymbol{\theta}^j_{k,-} : k \in \mathcal{K})$ are the parameters of the target DQNs of all agents at some previous decision epoch before epoch j. The gradient for each agent $k \in \mathcal{K}$ can be calculated as in (18). We summarize the online implementation of our proposed Deep-SARL algorithm for solving the stochastic computation offloading in a MEC system as in Algorithm 2.

$$L_{(\mathrm{DARLING})}\big(\boldsymbol{\theta}^j\big) = \tag{12}$$
$$\mathsf{E}_{(\boldsymbol{\chi},(c,e),u(\boldsymbol{\chi},(c,e)),\boldsymbol{\chi}')\in\widetilde{\mathcal{M}}^j}\left[\left((1-\gamma)\cdot u(\boldsymbol{\chi},(c,e)) + \gamma\cdot Q\left(\boldsymbol{\chi}', \arg\max_{(c',e')} \mathbb{Q}\big(\boldsymbol{\chi}',(c',e');\boldsymbol{\theta}^j\big);\boldsymbol{\theta}^j_-\right) - Q\big(\boldsymbol{\chi},(c,e);\boldsymbol{\theta}^j\big)\right)^2\right]$$

$$\nabla_{\boldsymbol{\theta}^j} L_{(\mathrm{DARLING})}\big(\boldsymbol{\theta}^j\big) = \tag{13}$$
$$\mathsf{E}_{(\boldsymbol{\chi},(c,e),u(\boldsymbol{\chi},(c,e)),\boldsymbol{\chi}')\in\widetilde{\mathcal{M}}^j}\left[\left((1-\gamma)\cdot u(\boldsymbol{\chi},(c,e)) + \gamma\cdot Q\left(\boldsymbol{\chi}', \arg\max_{(c',e')} Q\big(\boldsymbol{\chi}',(c',e');\boldsymbol{\theta}^j\big);\boldsymbol{\theta}^j_-\right) - Q\big(\boldsymbol{\chi},(c,e);\boldsymbol{\theta}^j\big)\right)\cdot \nabla_{\boldsymbol{\theta}^j} Q\big(\boldsymbol{\chi},(c,e);\boldsymbol{\theta}^j\big)\right]$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2876279, IEEE Internet of Things Journal

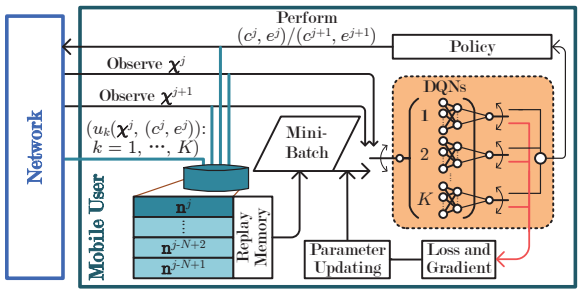


Fig. 3. Deep SARSA reinforcement learning (Deep-SARL) based stochastic computation offloading in a mobile-edge computing system.

Algorithm 2 Online Deep-SARL Algorithm for Stochastic Computation Task Offloading in A MEC System

1: **input** The DQNs and the target DQNs with two sets $\boldsymbol{\theta}^j = (\boldsymbol{\theta}^j_k : k \in \mathcal{K})$ and $\boldsymbol{\theta}^j_- = (\boldsymbol{\theta}^j_{k,-} : k \in \mathcal{K})$ of random parameters, for $j = 1$.
2: **initialize** for $j = 1$, the replay memory $\mathcal{N}^j$ with a finite size of $N \in \mathbb{N}_+$, the mini-batch $\widetilde{\mathcal{N}}^j$ with a size of $\tilde{N} < N$ for experience replay, the initial network state $\boldsymbol{\chi}^j \in \mathcal{X}$, a joint control action $(c^j, e^j) \in \mathcal{Y}$ randomly with probability ϵ or $(c^j, e^j) = \arg\max_{(c,e)\in\mathcal{Y}} \sum_{k\in\mathcal{K}} Q_k(\boldsymbol{\chi}^j, (c, e); \boldsymbol{\theta}^j_k)$ with probability $1 - \epsilon$.
3: **repeat**
4: After performing the selected joint control action (c^j, e^j), the agents realize immediate utilities $(u_k(\boldsymbol{\chi}^j, (c^j, e^j)) : k \in \mathcal{K})$.
5: The MU observes the new network state $\boldsymbol{\chi}^{j+1} \in \mathcal{X}$ at the next epoch $j + 1$, which is taken as an input to the DQNs of all agents with parameters $\boldsymbol{\theta}^j$, and selects a joint control action $(c^{j+1}, e^{j+1}) \in \mathcal{Y}$ randomly with probability ϵ or $(c^{j+1}, e^{j+1}) = \arg\max_{(c,e)\in\mathcal{Y}} \sum_{k\in\mathcal{K}} Q_k(\boldsymbol{\chi}^{j+1}, (c, e); \boldsymbol{\theta}^j_k)$ with probability $1 - \epsilon$.
6: The replay memory $\mathcal{N}^j$ is updated with the most recent transition $\mathbf{n}^j = (\boldsymbol{\chi}^j, (c^j, e^j), (u_k(\boldsymbol{\chi}^j, (c^j, e^j)) : k \in \mathcal{K}), \boldsymbol{\chi}^{j+1}, (c^{j+1}, e^{j+1}))$.
7: With a randomly sampled mini-batch of transitions $\widetilde{\mathcal{N}}^j \subseteq \mathcal{N}^j$, all agents update the DQN parameters $\boldsymbol{\theta}^j$ using the gradient as in (18).
8: The target DQNs are regularly reset by $\boldsymbol{\theta}^{j+1}_- = \boldsymbol{\theta}^j$, and otherwise $\boldsymbol{\theta}^{j+1}_- = \boldsymbol{\theta}^j_-$.
9: The decision epoch index is updated by $j \leftarrow j + 1$.
10: **until** A predefined stopping condition is satisfied.
11: **return** The parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_k : k \in \mathcal{K})$ associated with the DQNs of all agents.

V. Numerical Experiments

In this section, we proceed to evaluate the stochastic computation offloading performance achieved from our derived online learning algorithms, namely, DARLING and Deep-SARL.

A. General Setup

Throughout the experiments, we suppose there are $B = 6$ BSs in the system connecting the MU with the MEC server. The channel gain states between the MU and the BSs are from a common finite set $\{-11.23, -9.37, -7.8, -6.3, -4.68, -2.08\}$ (dB), the transitions of which happen across the discrete decision epochs following respective randomly generated matrices. Each energy unit corresponds to $2 \cdot 10^{-3}$ Joule, and the number of energy units received from the wireless environment follows a Poisson arrival process with average arrival rate $\lambda_{(\mathrm{e})}$ (in units per epoch). We set $K = 5$ for the Deep-SARL algorithm, while the $u^{(1)}\big(\boldsymbol{\chi}^j, (c^j, e^j)\big)$, $u^{(2)}\big(\boldsymbol{\chi}^j, (c^j, e^j)\big)$, $u^{(3)}\big(\boldsymbol{\chi}^j, (c^j, e^j)\big)$, $u^{(4)}\big(\boldsymbol{\chi}^j, (c^j, e^j)\big)$ and $u^{(5)}\big(\boldsymbol{\chi}^j, (c^j, e^j)\big)$ in (6) are chosen to be the exponential functions. Then,

$$\begin{aligned}
u_1\big(\boldsymbol{\chi}^j, (c^j, e^j)\big) &= \omega_1 \cdot u^{(1)}\big(\boldsymbol{\chi}^j, (c^j, e^j)\big) \\
&= \omega_1 \cdot \exp\big\{-\min\big\{d^j, \delta\big\}\big\}, \\
u_2\big(\boldsymbol{\chi}^j, (c^j, e^j)\big) &= \omega_2 \cdot u^{(2)}\big(\boldsymbol{\chi}^j, (c^j, e^j)\big) \\
&= \omega_2 \cdot \exp\big\{-\eta^j\big\}, \\
u_3\big(\boldsymbol{\chi}^j, (c^j, e^j)\big) &= \omega_3 \cdot u^{(3)}\big(\boldsymbol{\chi}^j, (c^j, e^j)\big) \\
&= \omega_3 \cdot \exp\big\{-\rho^j\big\}, \\
u_4\big(\boldsymbol{\chi}^j, (c^j, e^j)\big) &= \omega_4 \cdot u^{(4)}\big(\boldsymbol{\chi}^j, (c^j, e^j)\big) \\
&= \omega_4 \cdot \exp\big\{-\varphi^j\big\}, \\
u_5\big(\boldsymbol{\chi}^j, (c^j, e^j)\big) &= \omega_5 \cdot u^{(5)}\big(\boldsymbol{\chi}^j, (c^j, e^j)\big) \\
&= \omega_5 \cdot \exp\big\{-\phi^j\big\}.
\end{aligned}$$

Based on the works [39] and [40], we use a single hidden layer consisting of 200 neurons[5] for the design of a DQN in DARLING algorithm and select tanh as the activation function [42] and Adam as the optimizer. The same number of 200 neurons are employed to design the single-layer DQNs of all agents in Deep-SARL and hence for each DQN of an agent, there are in total 40 neurons. Both the DARLING and the Deep-SARL algorithms are experimented in TensorFlow [27]. Other parameter values used in experiments are listed in Table I.

For the purpose of performance comparisons, we simulate three baselines as well, namely,

1) *Mobile Execution* – The MU processes all scheduled computation task at the mobile device with maximum possible energy, that is, at each decision epoch j, $c^j = 0$ and

$$e^j = \begin{cases} \min\Big\{q^j_{(\mathrm{e})}, \Big\lfloor \Big(f^{(\max)}_{(\mathrm{CPU})}\Big)^3 \cdot \tau \Big\rfloor\Big\}, & \text{if } q^j_{(\mathrm{e})} > 0; \\ 0, & \text{if } q^j_{(\mathrm{e})} = 0, \end{cases}$$

[5]The tradeoff between a time demanding training process and an improvement in performance with a deeper and/or wider neural network is still an open problem [41].

$$Q^{j+1}_k(\boldsymbol{\chi}, (c, e)) = Q^j_k(\boldsymbol{\chi}, (c, e)) + \alpha^j \cdot \Big((1 - \gamma) \cdot u_k(\boldsymbol{\chi}, (c, e)) + \gamma \cdot Q^j_k(\boldsymbol{\chi}', (c', e')) - Q^j_k(\boldsymbol{\chi}, (c, e))\Big) \tag{16}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2876279, IEEE Internet of Things Journal

TABLE I
PARAMETER VALUES IN EXPERIMENTS.

Parameter	Value
Replay memory capacities M, N	5000, 5000
Mini-batch sizes $\tilde{M}$, $\tilde{N}$	200, 200
Decision epoch duration δ	$5 \cdot 10^{-3}$ second
Channel bandwidth W	0.6 MHz
Noise power I	$1.5 \cdot 10^{-8}$ Watt
Input data size μ	10^4 bits
CPU cycles ν	$7.375 \cdot 10^6$
Maximum CPU-cycle frequency $f_{(\mathrm{CPU})}^{(\max)}$	2×10^9 Hz
Maximum transmit power $p_{(\mathrm{tr})}^{(\max)}$	2 Watt
Handover delay ζ	2×10^{-3} second
MEC service price ξ	1
Weights ω_1, ω_2, ω_3, ω_4, ω_5	3, 9, 5, 2, 1
Maximum task queue length $q_{(\mathrm{t})}^{(\max)}$	4 tasks
Maximum energy queue length $q_{(\mathrm{e})}^{(\max)}$	4 units
Discount factor γ	0.9
Exploration probability ϵ	0.01

where the allocation of energy units takes into account the maximum CPU-cycle frequency $f_{(\mathrm{CPU})}^{(\max)}$ and $\lfloor \cdot \rfloor$ means the floor function.

2) *Server Execution* – According to Lemma 2, with maximum possible energy units in the energy queue that satisfy the maximum transmit power constraint, the MU always selects a BS that achieves the minimum task execution delay to offload the input data of a scheduled computation task for MEC server execution.
3) *Greedy Execution* – At each decision epoch, the MU decides to execute a computation task at its own mobile device or offload it to the MEC server for processing with the aim of minimizing the immediate task execution delay.

B. Experiment Results

We carry out experiments under various settings to validate the proposed studies in this paper.

1) Experiment 1 – Convergence performance: Our goal in this experiment is to validate the convergence property of our proposed algorithms, namely, DARLING and Deep-SARL, for stochastic computation offloading in the considered MEC system. We set the task arrival probability and the average energy arrival rate to be $\lambda_{(\mathrm{t})} = 0.5$ and $\lambda_{(\mathrm{e})} = 0.8$ units per epoch, respectively. Without loss of the generality, we plot the simulated variations in $Q\big(\boldsymbol{\chi}, (c,e); \boldsymbol{\theta}^j\big)$ (where $\boldsymbol{\chi} = (2,2,2,(-6.3,-6.3,-4.68,-7.8,-6.3,-6.3))$, $(c,e) = (2,4)$ and $j \in \mathbb{N}_+$) for the DARLING algorithm as well as the loss function defined by (17) for the Deep-SARL algorithm versus the decision epochs in Fig. 4, which reveals that the convergence behaviours of both DARLING and Deep-SARL are similar. The two algorithms converge at a reasonable speed, for which around 1.5×10^4 decision epochs are needed.

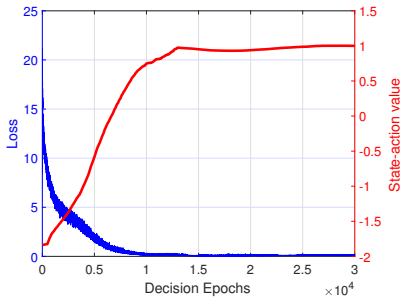


Fig. 4. Illustration for the convergence property of DARLING and Deep-SARL.

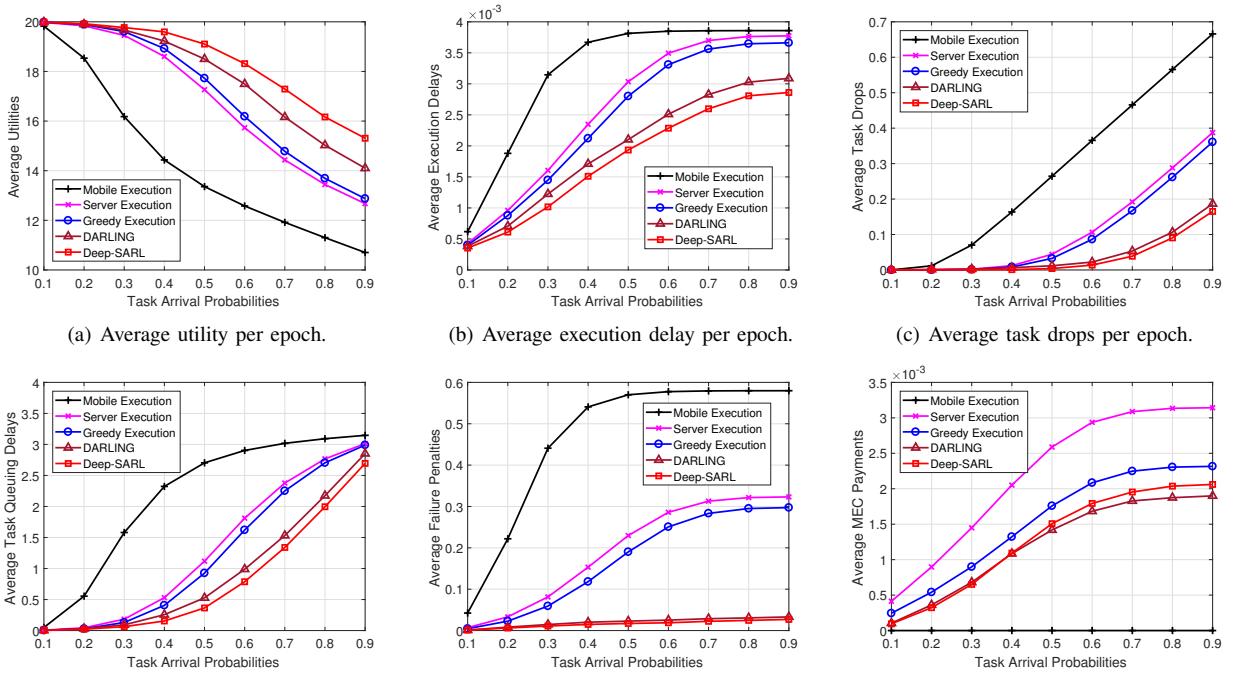
2) Experiment 2 – Performance under various $\lambda_{(\mathrm{t})}$: In this experiment, we try to demonstrate the average computation offloading performance per epoch in terms of the average utility, the average task execution delay, the average task drops, the average task queuing delay, the average task failure penalty and the average MEC service payment under different computation task arrival probability settings. We choose for the average energy unit arrival rate as $\lambda_{(\mathrm{e})} = 1.6$ units per epoch. The results are exhibited in Fig. 5. Fig. 5 (a) illustrates the average utility performance when the MU implements DARLING and Deep-SARL. Figs. 5 (b)–(f) illustrate the average task execution delay, the average task drops, the average task queuing delay, the average task failure penalty and the average MEC service payment.

Each plot compares the performance of the DARLING and the Deep-SARL to the three baseline computation offloading schemes. From Fig. 5, it can be observed that both the proposed schemes achieve a significant gain in average utility. Similar observations can be made from the curves in other plots, though the average MEC service payment per epoch from Deep-SARL is a bit higher than that from DARLING. This can be explained by the reason that with the network settings by increasing the task arrival probability in this experiment, more tasks are scheduled for execution at the MEC server using the Deep-SARL algorithm. As the computation task arrival probability increases, the average utility performances decrease due to the increases in average

$$L_{(\mathrm{Deep-SARL})}\big(\boldsymbol{\theta}^j\big) = \tag{17}$$
$$\mathsf{E}_{(\boldsymbol{\chi},(c,e),(u_k(\boldsymbol{\chi},(c,e)):k\in\mathcal{K}),\boldsymbol{\chi}',(c',e'))\in\widetilde{\mathcal{N}}^j}\left[\sum_{k\in\mathcal{K}}\Big((1-\gamma)\cdot u_k(\boldsymbol{\chi},(c,e)) + \gamma\cdot Q_k\Big(\boldsymbol{\chi}',(c',e');\boldsymbol{\theta}^j_{k,-}\Big) - Q_k\Big(\boldsymbol{\chi},(c,e);\boldsymbol{\theta}^j_k\Big)\Big)^2\right]$$

$$\nabla_{\boldsymbol{\theta}^j_k} L_{(\mathrm{Deep-SARL})}\big(\boldsymbol{\theta}^j\big) = \tag{18}$$
$$\mathsf{E}_{(\boldsymbol{\chi},(c,e),(u_k(\boldsymbol{\chi},(c,e)):k\in\mathcal{K}),\boldsymbol{\chi}',(c',e'))\in\widetilde{\mathcal{N}}^j}\left[\begin{array}{c}\Big((1-\gamma)\cdot u_k(\boldsymbol{\chi},(c,e)) + \gamma\cdot Q_k\Big(\boldsymbol{\chi}',(c',e');\boldsymbol{\theta}^j_{k,-}\Big) - Q_k\Big(\boldsymbol{\chi},(c,e);\boldsymbol{\theta}^j_k\Big)\Big)\cdot \\ \nabla_{\boldsymbol{\theta}^j_k} Q_k\Big(\boldsymbol{\chi},(c,e);\boldsymbol{\theta}^j_k\Big)\end{array}\right]$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2876279, IEEE Internet of Things Journal



(a) Average utility per epoch. (b) Average execution delay per epoch. (c) Average task drops per epoch.

(d) Average task queuing delay per epoch. (e) Average task failure penalty per epoch. (f) Average MEC service payment per epoch.

Fig. 5. Average computation offloading performance versus task arrival probabilities.

task execution delay, average task drops, average task queuing delay, average task failure penalty and average MEC service payment. Since there are not enough energy units in the energy queue during one decision epoch on average, in order to avoid task drops and task failure penalty, only a portion of the queued energy units are allocated for processing a scheduled computation task, hence leading to more queued tasks, i.e., an increased average task queueing delay. The utility performance from Deep-SARL outperforms that from DARLING. This indicates that by combining a deep neural network and the Q-function decomposition technique, the original stochastic computation offloading problem becomes simplified, hence performance improvement can be expected from approximating the state-action Q-functions of all agents with the same number of neurons.

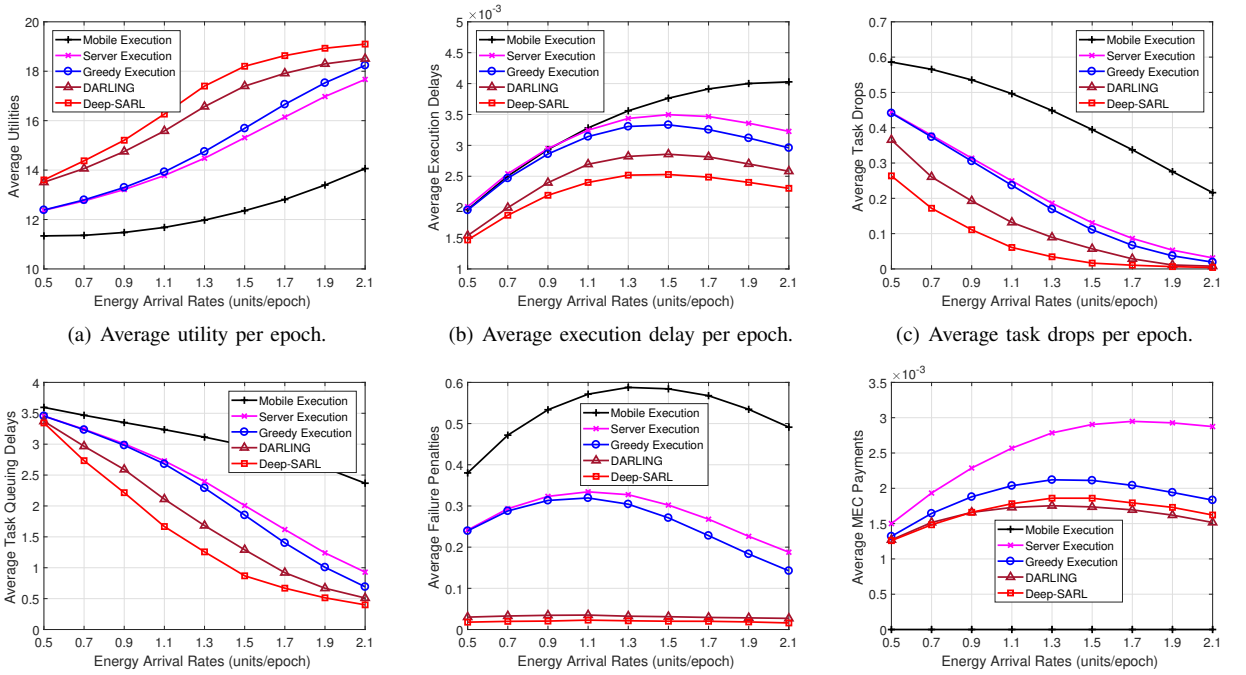
3) Experiment 3 – Performance with changing $\lambda_{(\mathrm{e})}$: We do the third experiment to simulate the average computation offloading performance per epoch achieved from the derived DARLING and Deep-SARL algorithms and other three baselines versus the average energy unit arrival rates. The computation task arrival probability in this experiment is set to be $\lambda_{(\mathrm{t})} = 0.6$. The per epoch average utility, average task execution delay, average task drops, average task queuing delay, average task failure penalty and average MEC service payment across the entire learning period are depicted in Fig. 6. We can clearly see from Fig. 6 that as the available energy units increase, the overall computation offloading performance improves. However, as the energy unit arrival rate increases, the average task execution delay, the average task failure penalty and the average MEC service payment[6] first increase but then decrease. Compared to ω_1, ω_4 and ω_5, we set larger values for ω_2 and ω_3. The priority of minimizing average task drops and average task queuing delay pushes the MU to schedule as many computation tasks for execution as possible. Hence the increasing number of queued energy units provides more opportunities to execute a computation task during each decision epoch, and at the same time, increases the possibility of failing to execute a task. When the average number of energy units in the energy queue increases to a sufficiently large value, enough energy units can be allocated to each scheduled computation task, due to which the task execution delay, the possibility of task computation failures and the MEC service payment decrease.

[6]It's easy to see that the mobile execution scheme does not use MEC service, hence no MEC service payment needs to be made.

VI. Conclusions

In this paper, we put our emphasis to investigate the design of a stochastic computation offloading policy for a representative MU in an ultra-dense sliced RAN by taking into account the dynamics generated from the time-varying channel qualities between the MU and the BSs, energy units received from the wireless environment as well as computation task arrivals. The problem of stochastic computation offloading is formulated as a MDP, for which we propose two double DQN-based online strategic computation offloading algorithms, namely, DARLING and Deep-SARL. Both learning algorithms survive the curse of high dimensionality in state space and need no a priori information of dynamics statistics. We find from numerical experiments that compared to three baselines, our derived algorithms can achieve much better long-term utility performance, which indicates an optimal tradeoff among the computation task execution delay, the task drops, the task queuing delay, the task failure penalty and the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2876279, IEEE Internet of Things Journal



(a) Average utility per epoch. (b) Average execution delay per epoch. (c) Average task drops per epoch.

(d) Average task queuing delay per epoch. (e) Average task failure penalty per epoch. (f) Average MEC service payment per epoch.

Fig. 6. Average computation offloading performance versus average energy unit arrival rates.

MEC service payment. Moreover, the Deep-SARL algorithm outperforms the DARLING algorithm by taking the advantage of the additive utility function structure.

ACKNOWLEDGEMENTS

This research was supported in part by National Key R&D Program of China under Grant 2018YFB0803702, National Natural Science Foundation of China under Grants 61701439 and 61731002, Zhejiang Key Research and Development Plan under Grant 2018C03056, National Science Foundation under Grant CNS-1702957, Wireless Engineering Research and Engineering Center (WEREC) at Auburn University, JSPS KAKENHI under Grant JP16H02817, and Academy of Finland under grant 289611.

APPENDIX A: PROOF OF LEMMA 1

With the decisions of computation offloading $c^j \in \mathcal{B}$ and energy unit allocation $e^j > 0$ at an epoch j, the input data transmission is independent of the network dynamics. Suppose there exists a new transmission policy with which the MU changes its transmission rate from $r^{j,(1)}$ to $r^{j,(2)} \neq r^{j,(1)}$ at a certain point during the task input data transmission. We denote the time durations corresponding to transmission rates $r^{j,(1)}$ and $r^{j,(2)}$ as $d^{j,(1)}_{(\mathrm{tr})}$ and $d^{j,(2)}_{(\mathrm{tr})}$, respectively. Taking into account the maximum transmit power of the mobile device (4), it is easy to verify that the following two constraints on total energy consumption:

$$\min\left\{e^j, \left(d^{j,(1)}_{(\mathrm{tr})} + d^{j,(2)}_{(\mathrm{tr})}\right)\cdot p^{(\max)}_{(\mathrm{tr})}\right\} = \frac{I}{g^j_{c^j}}\cdot\left(2^{\frac{r^{j,(1)}}{W}} - 1\right)\cdot d^{j,(1)}_{(\mathrm{tr})} + \frac{I}{g^j_{c^j}}\cdot\left(2^{\frac{r^{j,(2)}}{W}} - 1\right)\cdot d^{j,(2)}_{(\mathrm{tr})},$$

and total transmitted data bits:

$$r^{j,(1)}\cdot d^{j,(1)}_{(\mathrm{tr})} + r^{j,(2)}\cdot d^{j,(2)}_{(\mathrm{tr})} = \mu,$$

can be satisfied. On the other hand, the average transmission rate within a duration of $d^{j,(1)}_{(\mathrm{tr})} + d^{j,(2)}_{(\mathrm{tr})}$ can be written as

$$\bar{r}^j = W\cdot\log_2\left(1 + \frac{g^j_{c^j}}{I}\cdot\frac{\min\left\{e^j, \left(d^{j,(1)}_{(\mathrm{tr})} + d^{j,(2)}_{(\mathrm{tr})}\right)\cdot p^{(\max)}_{(\mathrm{tr})}\right\}}{d^{j,(1)}_{(\mathrm{tr})} + d^{j,(2)}_{(\mathrm{tr})}}\right).$$

Consider using $\bar{r}^j$ as the only transmission rate during the whole period of task input data transmission, we construct the deduction as in (19), where the inequality originates from the concavity of a logarithmic function. From (19), we can find that with a constant transmission rate, more data can be transmitted within the same transmission period. Hence, applying a constant rate for the transmission of μ task input data bits achieves the minimum transmission time, which can be solved according to (2) and (3). This completes the proof.

APPENDIX B: PROOF OF LEMMA 2

By replacing r^j in (2) with (3), we get

$$\log_2\left(1 + I^{-1}\cdot g^j_{c^j}\cdot e^j\cdot\frac{1}{d^j_{(\mathrm{tr})}}\right) = \frac{\mu}{W}\cdot\frac{1}{d^j_{(\mathrm{tr})}}. \quad (20)$$

Alternatively, we take $1/d^j_{(\mathrm{tr})}$ as the solution of (20), which is an intersection of two lines, namely, $\ell_1(x) = \log_2(1 + g^j_{c^j}\cdot I^{-1}\cdot e^j\cdot x)$ and $\ell_2(x) = \frac{\mu}{W}\cdot x$, for $x > 0$. From the non-negativity and the monotonicity of a logarithmic function and a linear function, it is easy to see that $1/d^j_{(\mathrm{tr})}$ is a monotonically

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2876279, IEEE Internet of Things Journal

increasing function of e^j. Thus, $d^j_{(\mathrm{tr})}$ is a monotonically decreasing function of e^j.

Appendix C: Proof of Theorem 1

For the state-action Q-function of a joint action $(c, e) \in \mathcal{Y}$ in a network state $\boldsymbol{\chi} \in \mathcal{X}$ as in (9), we have (21), which completes the proof.

Appendix D: Proof of Theorem 2

Since the per-agent state-action Q-functions are learned simultaneously, we consider the monolithic updates during the learning process of the SARL algorithm, namely, the updating rule in (16) can be then encapsulated as

$$\sum_{k\in\mathcal{K}} Q_k^{j+1}(\boldsymbol{\chi},(c,e)) = \left(1-\alpha^j\right)\cdot \sum_{k\in\mathcal{K}} Q_k^{j}(\boldsymbol{\chi},(c,e)) + \quad (22)$$
$$\alpha^j\cdot\left((1-\gamma)\cdot\sum_{k\in\mathcal{K}} u_k(\boldsymbol{\chi},(c,e)) + \gamma\cdot\sum_{k\in\mathcal{K}} Q_k^{j}(\boldsymbol{\chi}',(c',e'))\right),$$

where $(\boldsymbol{\chi},(c,e)),\ (\boldsymbol{\chi}',(c',e')) \in \mathcal{X}\times\mathcal{Y}$. We rewrite (22) as

$$\sum_{k\in\mathcal{K}} Q_k^{j+1}(\boldsymbol{\chi},(c,e)) - \sum_{k\in\mathcal{K}} Q_k(\boldsymbol{\chi},(c,e)) = (1-\alpha^j)\cdot \quad (23)$$
$$\left(\sum_{k\in\mathcal{K}} Q_k^{j}(\boldsymbol{\chi},(c,e)) - \sum_{k\in\mathcal{K}} Q_k(\boldsymbol{\chi},(c,e))\right) + \alpha^j\cdot\Upsilon^j(\boldsymbol{\chi},(c,e)),$$

where

$$\Upsilon^j(\boldsymbol{\chi},(c,e)) = (1-\gamma)\cdot\sum_{k\in\mathcal{K}} u_k(\boldsymbol{\chi},(c,e)) +$$
$$\gamma\cdot\max_{(c'',e'')}\sum_{k\in\mathcal{K}} Q_k^{j}(\boldsymbol{\chi}',(c'',e'')) - \sum_{k\in\mathcal{K}} Q_k(\boldsymbol{\chi},(c,e))) +$$
$$\gamma\cdot\left(\sum_{k\in\mathcal{K}} Q_k^{j}(\boldsymbol{\chi}',(c',e')) - \max_{(c'',e'')}\sum_{k\in\mathcal{K}} Q_k^{j}(\boldsymbol{\chi}',(c'',e''))\right).$$

Let $\mathcal{O}^j = \sigma(\{(\boldsymbol{\chi}^z,(c^z,e^z),(u_k(\boldsymbol{\chi}^z,(c^z,e^z)) : k\in\mathcal{K})) : z\le j\},(Q_k^j(\boldsymbol{\chi},(c,e)) : \boldsymbol{\chi}\in\mathcal{X},(c,e)\in\mathcal{Y},k\in\mathcal{K}))$ denote the learning history for the first j decision epochs. The per-agent state-action Q-functions are $\mathcal{O}^j$-measurable, thus the $(\sum_{k\in\mathcal{K}} Q_k^{j+1}(\boldsymbol{\chi},(c,e)) - \sum_{k\in\mathcal{K}} Q_k(\boldsymbol{\chi},(c,e)))$ and the $\Upsilon^j(\boldsymbol{\chi},(c,e))$ are both $\mathcal{O}^j$-measurable. We then arrive at (24), where (a) is due to the convergence property of a standard Q-learning [22]. We are left with verifying that $\|\mathsf{E}[\gamma\cdot(\sum_{k\in\mathcal{K}} Q_k^j(\boldsymbol{\chi}',(c',e')) - \max_{(c'',e'')}\sum_{k\in\mathcal{K}} Q_k^j(\boldsymbol{\chi}',(c'',e'')))|\mathcal{O}^j]\|_\infty$ converges to zero with probability 1, which in our considered scenario follows from: i) an ϵ-greedy strategy is applied in SARL for choosing joint control actions; ii) the per-agent state-action Q-functions are upper bounded; and iii) both the network state and the joint control action spaces are finite. All conditions in [38, Lemma 1] are satisfied. Therefore, the convergence of the SARL learning process is ensured.

References

[1] "Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021," White Paper, Cisco, Feb. 2017.

[2] M. Satyanarayanan, "The emergence of edge computing," *IEEE Comput.*, vol. 50, no. 1, pp. 30–39, Jan. 2017.

[3] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong, "Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing," *J. Netw. Comput. Appl.*, vol. 59, pp. 46–54, Jun. 2016.

[4] Z. Su, Q. Xu, J. Luo, H. Pu, Y. Peng, and R. Lu, "A secure content caching scheme for disaster backup in fog computing enabled mobile social networks," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4579–4589, Oct. 2018.

[5] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, Q4 2017.

[6] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, Q3 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] 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.

[9] 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.

[10] X. Costa-Péerez, J. Swetina, T. Guo, R. Mahindra, and S. Rangarajan, "Radio access network virtualization for future mobile carrier networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 27–35, Jul. 2013.

[11] R. Kokku, R. Mahindra, H. Zhang, and S. Rangarajan, "NVS: A substrate for virtualizing wireless resources in cellular networks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1333–1346, Oct. 2012.

[12] X. Chen, Z. Han, H. Zhang, G. Xue, Y. Xiao, and M. Bennis, "Wireless resource scheduling in virtualized radio access networks using stochastic learning," *IEEE Trans. Mobile Comput.*, vol. 17, no. 4, pp. 961–974, Apr. 2018.

[13] P. Zhao, H. Tian, S. Fan, and A. Paulraj, "Information prediction and dynamic programming based RAN slicing for mobile edge computing," *IEEE Wireless Commun. Lett.*, Early Access Article, 2018.

[14] Y. Zhou, F. R. Yu, J. Chen, and Y. Kuo, "Resource allocation for information-centric virtualized heterogeneous networks with in-network caching and mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 11339–11351, Dec. 2017.

[15] C. You, K. Huang, H. Chae, and B. H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[16] X. Lyu, H. Tian, W. Ni, Y. Zhang, P. Zhang, and R. P. Liu, "Energy-efficient admission of delay-sensitive tasks for mobile edge computing," *IEEE Trans. Commun.*, Early Access Article, 2018.

[17] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, 'Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE ISIT*, Barcelona, Spain, Jul. 2016.

[18] Y. Mao, J. Zhang, 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.

[19] 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 WKSHP*, Singapore, Dec. 2017.

$$\bar{r}^j\cdot\left(d^{j,(1)}_{(\mathrm{tr})}+d^{j,(2)}_{(\mathrm{tr})}\right) = W\cdot\log_2\left(1+\frac{\left(2^{\frac{r^{j,(1)}}{W}}-1\right)\cdot d^{j,(1)}_{(\mathrm{tr})}+\left(2^{\frac{r^{j,(2)}}{W}}-1\right)\cdot d^{j,(2)}_{(\mathrm{tr})}}{d^{j,(1)}_{(\mathrm{tr})}+d^{j,(2)}_{(\mathrm{tr})}}\right)\cdot\left(d^{j,(1)}_{(\mathrm{tr})}+d^{j,(2)}_{(\mathrm{tr})}\right)$$
$$\ge\left(r^{j,(1)}\cdot\frac{d^{j,(1)}_{(\mathrm{tr})}}{d^{j,(1)}_{(\mathrm{tr})}+d^{j,(2)}_{(\mathrm{tr})}}+r^{j,(2)}\cdot\frac{d^{j,(2)}_{(\mathrm{tr})}}{d^{j,(1)}_{(\mathrm{tr})}+d^{j,(2)}_{(\mathrm{tr})}}\right)\cdot\left(d^{j,(1)}_{(\mathrm{tr})}+d^{j,(2)}_{(\mathrm{tr})}\right) = r^{j,(1)}\cdot d^{j,(1)}_{(\mathrm{tr})}+r^{j,(2)}\cdot d^{j,(2)}_{(\mathrm{tr})} = \mu \quad (19)$$

[20] Z. Jiang and S. Mao, "Energy delay tradeoff in cloud offloading for multi-core mobile devices," *IEEE Access*, vol. 3, pp. 2306–2316, Nov. 2015.

[21] 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, Jul. 2017.

[22] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3–4, pp. 279–292, May 1992.

[23] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[24] M. Abu Alsheikh, D. Niyato, S. Lin, H.-P. Tan, and D. I. Kim, "Fast adaptation of activity sensing policies in mobile devices," *IEEE Trans. Veh. Technol.*, vol. 66, no. 7, pp. 5995–6008, Jul. 2017.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[26] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI*, Phoenix, AZ, Feb. 2016.

[27] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *Proc. OSDI*, Savannah, GA, Nov. 2016.

[28] T. D. Burd and R. W. Brodersen, "Processor design for portable systems," *J. VLSI Signal Process. Syst.*, vol. 13, no. 2–3, pp. 203–221, Aug. 1996.

[29] X. Chen, P. Liu, H. Liu, C. Wu, and Y. Ji, "Multipath transmission scheduling in millimeter wave cloud radio access networks," in *Proc. IEEE ICC*, Kansas City, MO, May 2018.

[30] J. Kwak, O. Choi, S. Chong, and P. Mohapatra, "Processor-network speed scaling for energy-delay tradeoff in smartphone applications," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1647–1660, Jun. 2016.

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

[32] D. Adelman and A. J. Mersereau, "Relaxations of weakly coupled stochastic dynamic programs," *Oper. Res.*, vol. 56, no. 3, pp. 712–727, Jan. 2008.

[33] K. Gai, M. Qiu, Z. Xiong, and M. Liu, "Optimal resource allocation using reinforcement learning for IoT content-centric services," *Appl. Soft Comput.*, vol. 70, pp. 12–21, Sep. 2018.

[34] K. Gai, M. Qiu, M. Liu, H. Zhao, "Smart resource allocation using reinforcement learning in content-centric cyber-physical systems," in *Proc. SmartCom*, Shenzhen, China, Dec. 2017.

[35] S. Russel and A. L. Zimdars, "Q-decomposition for reinforcement learning agents," in *Proc. ICML*, Washington DC, Aug. 2003.

[36] L.-J. Lin, "Reinforcement learning for robots using neural networks," Carnegie Mellon University, 1992.

[37] G. A. Rummery and M. Niranjan. *Online Q-learning using connectionist systems*, Tech. Rep. CUED/F-INFENG/TR 166, Cambridge University Engineering Department, Sep. 1994.

[38] S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvári, "Convergence results for single-step on-policy reinforcement-learning algorithms," *Mach. Learn.*, vol. 38, no. 3, pp. 287–308, Mar. 2000.

[39] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Performance optimization in mobile-edge computing via deep reinforcement learning," *arXiv*, Mar. 2018.

[40] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, Las Vegas, NV, Jun. 2016.

[41] C. L. P. Chen and Z. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 10–24, Jan. 2018.

[42] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proc. IEEE ICCV*, Kyoto, Japan, Sep.–Oct. 2009.

Xianfu Chen received his Ph.D. degree in Signal and Information Processing, from the Department of Information Science and Electronic Engineering (ISEE) at Zhejiang University, Hangzhou, China, in March 2012. Since April 2012, he has been with the VTT Technical Research Centre of Finland Ltd, Oulu, Finland, where he is currently a Senior Scientist. His research interests cover various aspects of wireless communications and networking, with emphasis on human-level and artificial intelligence for resource awareness in next-generation communication networks. He is an IEEE member.

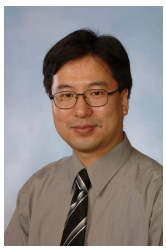
$$
\begin{aligned}
Q(\boldsymbol{\chi},(c,e)) &= \mathsf{E}_{\boldsymbol{\Phi}^*}\left[(1-\gamma)\cdot\sum_{j=1}^{\infty}(\gamma)^{j-1}\cdot u\big(\boldsymbol{\chi}^j,\big(c^j,e^j\big)\big)\,|\boldsymbol{\chi}^1=\boldsymbol{\chi},\big(c^1,e^1\big)=(c,e)\right]\\
&= \mathsf{E}_{\boldsymbol{\Phi}^*}\left[(1-\gamma)\cdot\sum_{j=1}^{\infty}(\gamma)^{j-1}\cdot\sum_{k\in\mathcal{K}}u_k\big(\boldsymbol{\chi}^j,\big(c^j,e^j\big)\big)\,|\boldsymbol{\chi}^1=\boldsymbol{\chi},\big(c^1,e^1\big)=(c,e)\right]\\
&= \sum_{k\in\mathcal{K}}\mathsf{E}_{\boldsymbol{\Phi}^*}\left[(1-\gamma)\cdot\sum_{j=1}^{\infty}(\gamma)^{j-1}\cdot u_k\big(\boldsymbol{\chi}^j,\big(c^j,e^j\big)\big)\,|\boldsymbol{\chi}^1=\boldsymbol{\chi},\big(c^1,e^1\big)=(c,e)\right]=\sum_{k\in\mathcal{K}}Q_k(\boldsymbol{\chi},(c,e))
\end{aligned}
\tag{21}
$$

$$
\begin{aligned}
&\left\|\mathsf{E}\left[\Upsilon^j(\boldsymbol{\chi},(c,e))\,|\mathcal{O}^j\right]\right\|_\infty \qquad (24)\\
&\le\left\|\mathsf{E}\left[(1-\gamma)\cdot\sum_{k\in\mathcal{K}}u_k(\boldsymbol{\chi},(c,e))+\gamma\cdot\max_{(c'',e'')}\sum_{k\in\mathcal{K}}Q_k^j(\boldsymbol{\chi}',(c'',e''))-\sum_{k\in\mathcal{K}}Q_k(\boldsymbol{\chi},(c,e))|\mathcal{O}^j\right]\right\|_\infty+\\
&\quad\left\|\mathsf{E}\left[\gamma\cdot\left(\sum_{k\in\mathcal{K}}Q_k^j(\boldsymbol{\chi}',(c',e'))-\max_{(c'',e'')}\sum_{k\in\mathcal{K}}Q_k^j(\boldsymbol{\chi}',(c'',e''))\right)|\mathcal{O}^j\right]\right\|_\infty\\
&\overset{(a)}{\le}\gamma\cdot\left\|\sum_{k\in\mathcal{K}}Q_k^j(\boldsymbol{\chi},(c,e))-\sum_{k\in\mathcal{K}}Q_k(\boldsymbol{\chi},(c,e))\right\|_\infty+\left\|\mathsf{E}\left[\gamma\cdot\left(\sum_{k\in\mathcal{K}}Q_k^j(\boldsymbol{\chi}',(c',e'))-\max_{(c'',e'')}\sum_{k\in\mathcal{K}}Q_k^j(\boldsymbol{\chi}',(c'',e''))\right)|\mathcal{O}^j\right]\right\|_\infty
\end{aligned}
$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2876279, IEEE Internet of Things Journal

Honggang Zhang is a Full Professor with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China. He is an Honorary Visiting Professor at the University of York, York, UK. He was the International Chair Professor of Excellence for Université Européenne de Bretagne (UEB) and Supélec, France. He served as the Chair of the Technical Committee on Cognitive Networks of the IEEE Communications Society from 2011 to 2012. He is currently active in the research on green communications and was the leading Guest Editor of the IEEE COMMUNICATIONS MAGAZINE special issues on "Green Communications". He was the co-author and an editor of two books with the titles of *Cognitive Communications Distributed Artificial Intelligence (DAI), Regulatory Policy and Economics, Implementation* (John Wiley & Sons) and *Green Communications: Theoretical Fundamentals, Algorithms and Applications* (CRC Press), respectively. He is an IEEE senior member.

Celimuge Wu received his PhD degree from The University of Electro-Communications, Japan in 2010, where he is currently an associate professor. His current research interests include vehicular ad hoc networks, IoT, 5G, and mobile cloud computing. He is/has been a TPC Co-Chair of Wireless Days 2019, ICT-DM 2018, and a track Co-Chair of many international conferences including ICCCN 2019, PIMRC 2016, IEEE ISC2 2017, ISNCC 2017, and WICON 2016. He is/has been serving as an associate editor of IEICE Transactions on Communications, a guest editor of IEEE Transactions on Emerging Topics in Computational Intelligence, IEEE Computational Intelligence Magazine, ACM/Springer MONET, MDPI Sensors, and Hindawi MIS. He is an IEEE member.



Shiwen Mao received a Ph.D. in electrical and computer engineering from Polytechnic University, Brooklyn, NY in 2004. He is the Samuel Ginn Distinguished Professor and Director of the Wireless Engineering Research and Education Center (WEREC) at Auburn University, Auburn, AL. His research interests include wireless networks and multimedia communications. He is a Distinguished Lecturer of the IEEE Vehicular Technology Society in the Class of 2014. He is on the Editorial Board of IEEE Transactions on Mobile Computing, IEEE Transactions on Multimedia, IEEE Internet of Things Journal, IEEE Multimedia, and ACM GetMobile, among others. He received the Auburn University Creative Research & Scholarship Award in 2018, the 2017 IEEE ComSoc ITC Outstanding Service Award, the 2015 IEEE ComSoc TC-CSR Distinguished Service Award, the 2013 IEEE ComSoc MMTC Outstanding Leadership Award, and the NSF CAREER Award in 2010. He is a co-recipient of the 2017 Best Conference Paper Award of IEEE ComSoc MMTC, IEEE SECON 2017 Best Demo Award, the Best Paper Awards from IEEE GLOBECOM 2016 & 2015, IEEE WCNC 2015, and IEEE ICC 2013, and the 2004 IEEE Communications Society Leonard G. Abraham Prize in the Field of Communications Systems. He is an IEEE senior member.

Yusheng Ji received her B.E., M.E., and D.E. degrees in electrical engineering from the University of Tokyo. She joined the National Center for Science Information Systems, Japan (NACSIS) in 1990. Currently, she is a Professor at the National Institute of Informatics, Japan (NII), and SOKENDAI (the Graduate University for Advanced Studies). Her research interests include network architecture, mobile computing, and network resource management. She is/has been an Editor of IEEE TVT, Associate Editor of IEICE Transactions and IPSJ Journal, Guest Editor-in-Chief, Guest Editor, and Guest Associate Editor of Special Issues of IEICE Transactions and IPSJ Journal, Symposium Co-chair of IEEE GLOBECOM 2012, 2014, Track Chair of IEEE VTC 2016 Fall, 2017 Fall, General Co-Chair of ICT-DM 2018, and a TPC member of IEEE INFOCOM, ICC, GLOBECOM, WCNC, VTC etc. She is an IEEE senior member.

Mehdi Bennis received his M.Sc. degree in Electrical Engineering jointly from the EPFL, Switzerland and the Eurecom Institute, France in 2002. From 2002 to 2004, he worked as a research engineer at IMRA-EUROPE investigating adaptive equalization algorithms for mobile digital TV. In 2004, he joined the Centre for Wireless Communications (CWC) at the University of Oulu, Finland as a research scientist. In 2008, he was a visiting researcher at the Alcatel-Lucent chair on flexible radio, SUPELEC. He obtained his Ph.D. in December 2009 on spectrum sharing for future mobile cellular systems. Currently Dr. Bennis is an Adjunct Professor at the University of Oulu and Academy of Finland research fellow. His main research interests are in radio resource management, heterogeneous networks, game theory and machine learning in 5G networks and beyond. He has co-authored one book and published more than 100 research papers in international conferences, journals and book chapters. He was the recipient of the prestigious 2015 Fred W. Ellersick Prize from the IEEE Communications Society and the 2016 Best Tutorial Prize from the IEEE Communications Society. Dr. Bennis serves as an editor for the IEEE Transactions on Wireless Communication. He is an IEEE senior member.