Reinforcement Learning-Based Mobile Offloading for Edge Computing Against Jamming and Interference

Liang Xiao, *Senior Member, IEEE*, Xiaozhen Lu, Tangwei Xu, *Student Member, IEEE*, Xiaoyue Wan, Wen Ji, *Senior Member, IEEE*, and Yanyong Zhang, *Fellow, IEEE*

***Abstract*— Mobile edge computing systems help improve the performance of computational-intensive applications on mobile devices and have to resist jamming attacks and heavy interference. In this paper, we present a reinforcement learning based mobile offloading scheme for edge computing against jamming attacks and interference, which uses safe reinforcement learning to avoid choosing the risky offloading policy that fails to meet the computational latency requirements of the tasks. This scheme enables the mobile device to choose the edge device, the transmit power and the offloading rate to improve its utility including the sharing gain, the computational latency, the energy consumption and the signal-to-interference-plus-noise ratio of the offloading signals without knowing the task generation model, the edge computing model, and the jamming/interference model. We also design a deep reinforcement learning based mobile offloading for edge computing that uses an actor network to choose the offloading policy and a critic network to update the actor network weights to improve the computational performance. We discuss the computational complexity and provide the performance bound that consists of the computational latency and the energy consumption based on the Nash equilibrium of the mobile offloading game. Simulation results show that this scheme can reduce the computational latency and save energy consumption.**

***Index Terms*— Mobile offloading, edge computing, interference, jamming, reinforcement learning.**

I. Introduction

With the booming of the computation-intensive applications such as face recognition and augmented reality (AR) games [1]–[3], mobile devices offload the computational tasks to the serving edge devices such as the neighboring access points (APs), base stations (BSs) and laptops to save energy consumption and reduce the computational latency. However, the offloading has to address jamming attacks with the goal to interrupt the ongoing offloading and interference from the neighboring radio devices. By applying the task allocation algorithms such as [4], a mobile device divides the task of the computing data into several parts and chooses the edge device, the offloading rate and the transmit power to reduce the computational latency, save energy consumption and increase the signal-to-interference-plus-noise ratio (SINR) of the offloading signals.

The optimal offloading policy that minimizes the computational latency and energy consumption, maximizes the SINR of the offloading signals, avoids activating reactive jammers and improves the interference control depends on the channel states, the computing model of the edge devices and the jamming strategy [1] and [5]. Nevertheless, most offloading optimization algorithms suffer from the unknown edge computing model and jamming/interference model, yielding higher computational latency and more energy consumption [6].

Reinforcement learning (RL) based mobile offloading enables mobile devices to choose the offloading rate without knowing the underlying edge computing model and network model [7]–[11]. In particular, a mobile offloading scheme named SEGUE as proposed in [7] that proposes a Q-learning based edge selection based on both the offloading history of the neighboring mobile devices and the radio bandwidth improves the offloading efficiency while its performance degrades in dynamic mobile edge computing (MEC) networks with strong jamming and/or heavy interference. The offloading scheme in [12] that applies the deep Q-network to choose both the edge device and the offloading rate suffers from serious performance degradation under smart jamming attacks and heavy interference.

In this paper, we design an RL based mobile offloading scheme for mobile edge computing that chooses the offloading policy to address jamming attacks and suppress interference. This scheme evaluates the quality of service (QoS) of the computational tasks such as the computational latency and quantizes the QoS into several levels as the risk levels.

Manuscript received January 17, 2020; revised May 2, 2020 and June 20, 2020; accepted June 30, 2020. Date of publication July 8, 2020; date of current version October 16, 2020. This work was supported in part by the Natural Science Foundation of China under Grant 61971366, in part by the National Key R&D Program of China (2017YFB1400100), in part by the Beijing Natural Science Foundation (4202072), and in part by the Key Research Program of Frontier Sciences, CAS (No. ZDBS-LY-JSC001). The associate editor coordinating the review of this article and approving it for publication was A. S. Cacciapuoti. *(Corresponding author: Liang Xiao.)*

Liang Xiao, Xiaozhen Lu, Tangwei Xu, and Xiaoyue Wan are with the Department of Information and Communication Engineering, Xiamen University, Xiamen 361005, China (e-mail: lxiao@xmu.edu.cn).

Wen Ji is with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China, and also with the Peng Cheng Laboratory, Shenzhen 518055, China (e-mail: jiwen@ict.ac.cn).

Yanyong Zhang is with the School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China (e-mail: yanyongz@ustc.edu.cn).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCOMM.2020.3007742

According to [13], the content popularity of the tasks is important for edge computing, as edge devices prefer to cache the computational tasks with higher content popularity and directly return the computational results to mobile devices with similar tasks to save their computational latency and energy consumption. Each mobile device applies the data analysis method in [14] to evaluate the popularity of the underlying task and evaluates the task sharing gain that represents the future reduced computational overhead of the edge device.

The offloading policy is chosen to save the energy consumption of the mobile device and reduce the task computational latency that consists of the local processing time, the offloading latency, the edge computational latency and feedback latency. The state that consists of the current content popularity, the previous energy consumption, the current radio bandwidth, the previous SINR of the offloading signals, and the previous task computational latency is the basis to update the Q-values.

We propose a deep RL based mobile offloading scheme for mobile devices that support deep learning to choose the offloading policy without being aware of the task generation model, the edge computing model and jamming/interference model. Instead of using a standard RL such as the actor-critic algorithm in the binary offloading scheme (SAC) in [15], this scheme designs a deep RL structure that consists of four deep neural networks to improve the mobile offloading performance and reduce the sample complexity. This scheme jointly optimizes the offloading rate and the transmit power besides the edge selection of SAC to reduce the task computational latency and save energy consumption. In this scheme, two online deep Q-networks (i.e., the actor network and the critic network) compress the continuous and high-dimensional action-state space, and two target networks (i.e., the target actor network and the target critic network) update the network weights with a slow tracking speed to improve the learning stability. More specifically, the actor network and critic network structures contain four fully connected layers rather than two fully connected layers to fully exploit the offloading experiences to improve the computational efficiency of mobile offloading in MEC. In addition, this scheme uses the state sequence as the input of the actor network to extract the temporal correlations between the states for higher computational efficiency. More specifically, an actor network is used to select the edge device, the transmit power and offloading rate instead of the offloading policy distribution or mixed policies from the neural network function approximators. A critic network is used to update the actor network weights for each offloading policy to reduce the exploration time for the mobile device.

We analyze the computational complexity of the proposed scheme and provide its performance bound including the task computational latency and the energy consumption based on the Nash equilibrium (NE) of the mobile offloading game, which has been ignored in previous works such as [15]. We perform simulations to verify the analysis, showing that the proposed scheme exceeds SEGUE in [7] with shorter computational latency, less energy consumption, and higher utility.

The rest of the paper is organized as follows. The related work is reviewed in Section II and the system model is presented in Section III. We propose the RL based mobile offloading scheme in Section IV and a deep RL based mobile offloading scheme for MEC in Section V. We then discuss the computational complexity and analyze the performance in Section VI. Simulation results are presented in Section VII, and conclusions are drawn in Section VIII.

Notation: Uppercase boldface letters denote matrices (e.g., $\mathbf{A}$), and lowercase boldface letters denote vectors (e.g., $\boldsymbol{a}$). $\mathbf{I}(\cdot)$ denotes the indicator function with value 1 if the argument is true, and 0 otherwise. $\nabla_{\boldsymbol{\theta}} f$ denotes the partial derivative of function f with respect to $\boldsymbol{\theta}$.

II. Related Work

Edge selection helps improve the computational and communication efficiency of the mobile device in edge computing. For example, in the MEC framework as presented in [16], the mobile device uses a semidefinite relaxation based offloading scheme to choose the central processing unit (CPU) frequency and allocate the computational task for shorter computational latency for given radio channel and edge computing model. The multi-server MEC system as developed in [17] presents a quasiconvex based edge selection to save energy consumption and decrease computational latency assuming that the user-edge channel follows the path-loss fading. The cooperative partial computational offloading algorithm in [18] applies convex optimization to select the edge device, the cloud server or the local CPU based on the known channel states to process the tasks following the task computational delay constraints.

As a typical MEC offloading rate optimization technique, a polynomial-time approximation based partial mobile offloading algorithm in [19] reduces the computational latency following the constraint of the resource utilization, the performance requirement, and the tree-structure tasks. The mobile offloading algorithm in [20] uses convex optimization to choose the offloading rate, the computational task allocation, and caching selection for quasistatic networks to save energy for latency-sensitive tasks. In the MEC system in [21], each user applies a Lagrange duality based offloading rate selection scheme based on the delay requirement and the channel states of all the users in the network. The energy-efficient edge computing framework as developed in [22] formulates an NP-hard problem to choose the offloading rate for mobile devices.

An example of the power control in MEC, the interference resistance algorithm as proposed in [5] uses graph coloring theory to choose the transmit power for shorter offloading latency under the 3GPP channel model for given interference levels in cellular networks. The resource allocation mechanism as designed in [23] uses Lyapunov optimization and matching theory to control the transmit power for each user to resist interference following the Rayleigh edge-user channel for the low-latency edge computing tasks. The MEC framework in [24] proposes a water-filling power allocation algorithm to improve the throughput of the Internet of Things based on the known channel state information.

TABLE I
SUMMARY OF THE MOBILE OFFLOADING METHODS

Offloading policy	Techniques	Assumptions	Interference suppression	Partial offloading	Ref
Edge selection	Semidefinite relaxation	Known channel states Known edge computing model	×	×	[16]
	Quasiconvex		✓	×	[17]
	Convex optimization		✓	✓	[18]
	Q-learning	Divided geographical regions	×	×	[7]
	Actor-critic mechanism	Binary offloading	×	×	[15]
Offloading rate	Polynomial-time approximation	Known channel states	×	✓	[19]
	Convex optimization		×	×	[20]
	Lagrange duality		×	✓	[21]
	Queuing theory		×	✓	[22]
Power control	Graph coloring theory	3GPP channel model Known interference levels	✓	×	[5]
	Lyapunov optimization Matching theory	Rayleigh edge-user channel	×	✓	[23]
	Water-filling algorithm	Known channel state information	✓	×	[24]
	Double deep Q-network	Stochastic computing model	×	×	[26]
BS working mode	Q-learning	Binary offloading	×	×	[25]

Reinforcement learning is promising to improve the computational performance. For example, the binary offloading based MEC framework as designed in [25] applies Q-learning to select the BS working modes for less energy consumption following the quality of service requirements. The virtual edge computing framework as designed in [26] applies double deep Q-network to choose the offloading edge and the transmit power to reduce the task queuing delay. The blockchain empowered binary offloading in [15] applies deep RL to choose the edge node for both the data mining and data processing tasks.

A mobile offloading scheme SEGUE as proposed in [7] that uses Q-learning to choose the edge device reduces the response time of the mobile offloading and satisfies the quality of service for the mobile device. Compared with SEGUE, this scheme improves the computational performance by incorporating the content popularity, the radio bandwidth, the computational latency, the previous transmission quality and the energy consumption in the mobile offloading against jamming attacks and interference. In addition, this work optimizes the offloading rate, the edge selection and the transmit power jointly, and proposes a deep RL based mobile offloading scheme to reduce the computational latency and save energy consumption. For ease of reference, the existing mobile offloading schemes are summarized in Table I.

III. SYSTEM MODEL

As shown in Fig. 1, we consider a mobile device such as a smartphone or a wearable device with limited battery life and computational resources that supports computation-intensive data-partition tasks such as face recognition and AR games [27]. The mobile device uses the task allocation algorithms to divide the generated task into several parts and chooses the offloading rate of the task to carry out the task faster than the existing binary offloading. The mobile device chooses one from the M edge devices such as the neighboring APs, BSs and laptops that have different computational speed, computational loading rate, and radio bandwidth.

The mobile device applies the task allocation algorithm such as [4] to divide the underlying data with $C^{(k)}$ bits in the task into X subtasks and chooses the offloading rate $a_2^{(k)} \in \{n/X | 0 \leq n \leq X\}$, where n is an integer. The content popularity of subtask j denoted by $\xi_{i,j}^{(k)} \in [0,1]$ that increases with its future reuse possibility on edge device i denoted by $\pi_{i,j}^{(k)}$ is evaluated according to the collaboration based greedy data analysis algorithm such as [14] based on the edge request history. The mobile device offloads $a_2^{(k)}X$ subtasks with the highest content popularity $[\xi_{a_1,j}^{(k)}]_{1\leq j\leq a_2^{(k)}X}$ to edge device $a_1^{(k)}$ with transmit power $a_3^{(k)}$, and processes the rest $(1-a_2^{(k)})X$ subtasks locally at l bps speed. Let $\mathbf{I}(\cdot)$ be the indicator function with value 1 if the argument is true, and 0 otherwise. The sharing gain of subtask j given by $C^{(k)}\mathbf{I}(\xi_{a_1,j}^{(k)} > 0)/X$ represents the future reduced computational latency of the subtask on edge device $a_1^{(k)}$, with $1 \leq j \leq a_2^{(k)}X$.

Upon receiving the offloading $a_2^{(k)}X$ subtasks, edge device i computes the subtasks at r_i bps speed and then returns the results via the feedback channel with bandwidth $\hat{b}_i^{(k)}$.

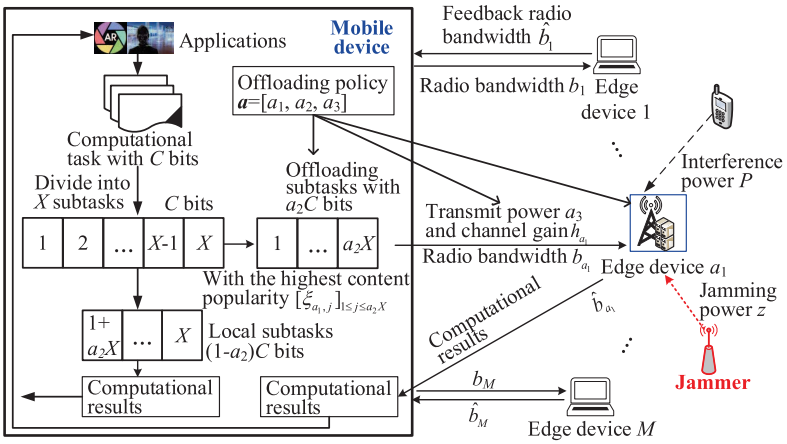


Fig. 1. Mobile offloading for a mobile device that offloads a_2C bits newly generated computational subtasks with transmit power a_3 to edge device a_1 and processes the rest of the computational task locally at time k, with $1 \leq a_1 \leq M$.

By integrating the local computational results and the edge feedback, the mobile device finds the task results. The feasible transmit power of the mobile device that ranges from zero to P_{T} is quantized into Ω levels, i.e., $a_3^{(k)} \in \{jP_{\mathrm{T}}/\Omega | 0 \leq j \leq \Omega\}$. According to [1], the receiver noise follows the Gusssian distribution with zero mean and standard deviation σ. The channel gain from the mobile device to edge device i is denoted by $h_i^{(k)}$ and the corresponding radio bandwidth is denoted by $b_i^{(k)}$, with $1 \leq i \leq M$. The mobile device is assumed to have ς effective capacitance compared with the edge device and has to use D CPU cycles to locally process each one-bit task.

Similar to [16], the task computational latency denoted by $t^{(k)}$ is the sum of the local processing time $\tau_{\mathrm{L}}^{(k)}$, the offloading latency $\tau_{\mathrm{T}}^{(k)}$, the edge computational latency $\tau_{\mathrm{E}}^{(k)}$, and the feedback latency $\tau_{\mathrm{F}}^{(k)}$. The energy consumption of the mobile device denoted by $\beta^{(k)}$ is the sum of the local computational energy consumption and the offloading energy consumption. The chosen edge device $a_1^{(k)}$ decodes the subtasks with $a_2^{(k)}C^{(k)}$ bits to measure the SINR of the offloading signals denoted by $\rho^{(k)}$, and then sends $\rho^{(k)}$ to the mobile device via the feedback channel.

The mobile offloading has to resist the interference with power $P^{(k)}$ and a jammer that aims to prevent the edge devices from receiving the offloading tasks. We consider a smart jammer that chooses the jamming power $z^{(k)}$ ranging from 0 to Y with unit jamming power cost υ based on the previous computational performance to minimize the utility of the mobile device with less jamming cost. Edge device i receives the jamming signals and interference signals, and quantizes the resulting interference power $y_i^{(k)}$ into L levels, i.e., $y_i^{(k)} \in \{jP_{\mathrm{J},i}^{\max}/L | 0 \leq j \leq L\}$. For simplicity, the jammer is assumed that cannot block the feedback channel of the mobile device. The important parameters are listed in Table II. Time index k will be omitted when no ambiguity results.

TABLE II
SUMMARY OF MAIN NOTATIONS

Parameters	Definitions
M	Number of the available edge devices
$a_1^{(k)} \in \{1, \ldots, M\}$	Selected edge device at time k
X	Number of the divided subtasks
$a_2^{(k)} \in \left\{\frac{n}{X}\right\}_{0 \leq n \leq X}$	Offloading rate of the mobile device
Ω	Number of the feasible transmit power levels
P_{T}	Maximum transmit power of the mobile device
$a_3^{(k)} \in \left\{\frac{jP_{\mathrm{T}}}{\Omega}\right\}_{0 \leq j \leq \Omega}$	Transmit power of the mobile device
$C^{(k)}$	Size of the computational tasks
D	Number of the CPU cycles for the mobile device to process one bit
l	Computational speed of the mobile device
r_i	Computational speed of edge device i
$b_i^{(k)}$	Radio bandwidth to edge device i
$\hat{b}_i^{(k)}$	Feedback bandwidth of edge device i
$h_i^{(k)}$	Channel gain to edge device i
$y_i^{(k)}$	Received jamming/interference power of edge device i
$\xi_{i,j}^{(k)}$	Content popularity of subtask j for edge device i
$t^{(k)}$	Computational latency of the tasks
$\beta^{(k)}$	Energy consumption of the mobile device
$\rho^{(k)}$	SINR of the offloading signals

IV. RL Based Mobile Offloading

We design an RL based mobile offloading scheme (SRLO) for MEC to select the edge device, the transmit power and the offloading rate based on the long-term risk level and Q-values to improve the task computational performance against smart jamming and interference. At time k, the mobile

device uses the task allocation algorithm such as [4] to divide the tasks into X subtasks, evaluates the content popularity $[\xi_{i,j}]_{1\le i\le M,1\le j\le X}$ of the X subtasks, and obtains the bandwidth $[b_i]_{1\le i\le M}$ to the M edge devices. The state denoted by $\boldsymbol{s}^{(k)}$ consists of the content popularity, the radio bandwidth, the task computational latency t, the energy consumption β and the SINR of the offloading signals ρ, i.e.,

$$\boldsymbol{s}^{(k)} = \left[[\xi_{i,j}]_{1\le i\le M,1\le j\le X}, [b_i]_{1\le i\le M}, t, \beta, \rho\right]. \quad (1)$$

According to [28], different types of tasks have different computational latency requirements. The task computational latency t is quantized into J levels to avoid exploring the dangerous offloading policy, and the level-j task has to be completed within η_j. More specifically, if the task computational latency t_j of the level-j task under $(\boldsymbol{s}^{(k)}, \boldsymbol{r})$ is larger than η_j, the risk level of $(\boldsymbol{s}^{(k)}, \boldsymbol{r})$ equals to 1, and 0 otherwise. This scheme uses the long-term risk level and the Q-values of each state-action pair. The long-term risk level relies on the future ψ risk levels and the risk discount factor denoted by γ that parametrizes the future reward degradation.

This offloading policy $\boldsymbol{a}^{(k)} = [a_\iota]_{1\le\iota\le 3}$ consists of the selected edge device a_1, the offloading rate a_2, and the transmit power a_3. The offloading policy set denoted by $\mathbf{A}$ is a $3\times M(X+1)(\Omega+1)$ matrix. The offloading policy distribution relies on the long-term risk level and the Q-values, and is given by

$$\Pr\left(\boldsymbol{a}^{(k)} = \boldsymbol{r}\right) = \frac{\exp\left(\frac{Q\left(\boldsymbol{s}^{(k)},\boldsymbol{r}\right)}{1+\sum_{\chi=0}^{\psi}\gamma^\chi \mathbf{I}\left(t_j(\boldsymbol{r})>\eta_j\right)}\right)}{\sum_{\tilde{\boldsymbol{a}}\in\mathbf{A}}\exp\left(\frac{Q\left(\boldsymbol{s}^{(k)},\tilde{\boldsymbol{a}}\right)}{1+\sum_{\chi=0}^{\psi}\gamma^\chi \mathbf{I}(t_j(\tilde{\boldsymbol{a}})>\eta_j)}\right)}. \quad (2)$$

The mobile device sends a_2X subtasks with a_2C bits to edge device a_1 with transmit power a_3 and processes the remaining $(1-a_2)X$ subtasks locally.

Upon receiving the feedback from the edge device, the mobile device measures the task computational latency t and the energy consumption β, and obtains the SINR of the previous offloading signal ρ. With the goal to increase the task sharing gain, save energy consumption, reduce the computational latency and increase the offloading SINR, the mobile device computes the utility with

$$u = \frac{C}{X}\sum_{j=1}^{a_2X}\mathbf{I}\left(\xi_{a_1,j}>0\right) - w_{\mathrm{L}}t - w_{\mathrm{E}}\beta + w_{\mathrm{S}}\rho \quad (3)$$

where w_{L}, w_{E} and w_{S} represent the importance of the computational latency, the energy consumption and the SINR of the offloading signals in the offloading process. The Q-values $\mathbf{Q}(\boldsymbol{s}^{(k)}, \boldsymbol{a}^{(k)})$ are then updated via the iterative Bellman equation based on the learning rate α and the discount rate δ according to [8].

Algorithm 1 RL based mobile offloading

Initialize $\gamma, \alpha, \delta, \psi, \rho, t, \beta$, and $\mathbf{Q} = \mathbf{0}$
for $k = 1, 2, \ldots$ **do**
  Divide the tasks into X subtasks
  Evaluate $[\xi_{i,j}]_{1\le i\le M,1\le j\le X}$
  Obtain $[b_i]_{1\le i\le M}$
  Formulate the state via (1)
  if k=1 **then**
    Randomly choose $\boldsymbol{a}^{(k)} \in \mathbf{A}$
  end if
  Update $\mathbf{Q}\left(\boldsymbol{s}^{(k)}, \boldsymbol{a}^{(k)}\right)$ via the iterative Bellman equation
  Choose $\boldsymbol{a}^{(k)}$ according to (2)
  Offload a_2X subtasks to edge device a_1 with transmit power a_3
  Process $(1-a_2)X$ subtasks locally
  Measure t and β
  Obtain the SINR ρ from edge device a_1
  Evaluate u via (3)
end for

V. Deep RL Based Mobile Offloading

We propose a deep RL based mobile offloading scheme (DDRLO) for mobile edge computing to further improve the computational performance. This scheme combines deep Q-network and an actor-critic algorithm to handle the continuous and high-dimensional offloading policies for the mobile device. Based on two online convolutional neural networks, i.e., the critic network and the actor network, this algorithm chooses the offloading policy instead of the offloading policy distribution or mixed policies. More specifically, the actor network chooses the edge device, the transmit power and the offloading rate based on the computational latency, the energy consumption, the SINR of the offloading signals, the content popularity of the X subtasks, and the radio bandwidth of the M edge devices, and the critic network updates the actor network weights. This algorithm uses two target networks to improve the learning stability during the offloading policy selection.

This scheme formulates the state similar to Algorithm 1 and forms a state sequence that consists of the previous λ offloading policies and states and the current state $\boldsymbol{s}^{(k)}$, i.e., $\{\boldsymbol{s}^{(\chi)}, \boldsymbol{a}^{(\chi)}, \boldsymbol{s}^{(k)}\}_{k-\lambda\le\chi\le k-1}$. The state sequence is input to the actor network, whose outputs are a three-dimensional vector $\boldsymbol{\omega}$. The offloading policy $\boldsymbol{a}^{(k)}$ is chosen based on $\boldsymbol{\omega}$ and the noise following the Ornstein-Uhlenbeck process $\mathcal{N}$. The mobile device offloads a_2C bits subtasks to edge device a_1 similar to Algorithm 1. Upon finishing the computational task, the mobile device evaluates the utility u via (3).

Both the actor network and the critic network have two convolutional (Conv.) layers and four fully connected (FC) layers, as shown in Fig. 2. The first Conv. layer of the two networks has a single input channel, and the Conv. layers are used to extract the mobile offloading features of the mobile device. The actor network has ϵ_1 filters in Conv. 1 and ϵ_2 filters in Conv. 2. Each filter in Conv. 1 and Conv. 2 has

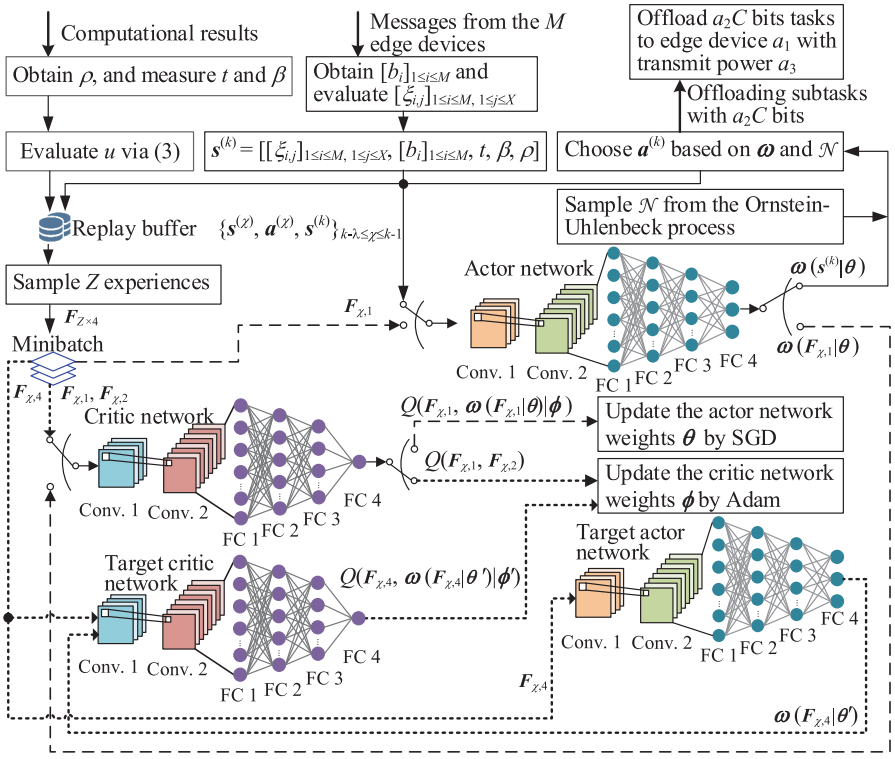


Fig. 2. Illustration of the deep RL based mobile offloading scheme.

n_1 kernels and n_2 kernels, respectively. The actor network uses g_1, g_2 and g_3 rectified linear units (ReLUs) in the three fully connected layers as the activation functions, which are used for mapping the features to the offloading policies. The fourth FC layer uses three sigmoid functions to output the three-dimensional vector $\boldsymbol{\omega}$. In the critic network, Conv. 1 has ε_1 filters, each with m_1 kernels. Conv. 2 has ε_2 filters, and each filter has m_2 kernels. The first three FC layers in the critic network use g_1', g_2' and g_3' ReLUs as the activation functions, respectively.

The mobile device saves the offloading experience as $\{\boldsymbol{s}^{(k-1)}, \boldsymbol{a}^{(k-1)}, u, \boldsymbol{s}^{(k)}\}$ to the replay buffer. The mobile device randomly samples Z offloading experiences as $\boldsymbol{F}_{Z\times 4}$ from the replay buffer to update the critic network weights denoted by ϕ and the actor network weights denoted by $\boldsymbol{\theta}$ in Fig. 2. By using Adam as the gradient descent algorithm [29], the critic network weights ϕ are updated similar to [30] with the minimized mean squared errors between the estimated Q-values and the optimal Q-value. The actor network weights $\boldsymbol{\theta}$ are updated by maximizing the sampled policy gradient with stochastic gradient descent (SGD) algorithm as follows,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\alpha}{Z}\sum_{\chi=1}^{Z}\nabla_{\boldsymbol{\theta}} Q\Big(\boldsymbol{F}_{\chi,1}, \boldsymbol{\omega}\big(\boldsymbol{F}_{\chi,1}|\boldsymbol{\theta}\big) \mid \phi\Big). \tag{4}$$

The mobile device uses a target actor network and a target critic network to avoid the instability exploration due to the correlations of state observations similar to [31], whose weights denoted by $\boldsymbol{\theta}'$ and ϕ' are soft updated with a learning factor $\kappa \ll 1$ to slow the tracking speed to the online actor network and critic network.

VI. Performance Evaluations

We analyze the computational complexity of the proposed schemes according to the analysis in [32] and [33]. The performance analysis regarding the task computational latency, the energy consumption and the utility is provided based on the Nash equilibrium of the mobile offloading game and the mobile edge computing model given by [16], [34] and [35].

Theorem 1: The computational complexity of the RL based mobile offloading is given by $O(kMX\Omega)$ and that of the deep RL based mobile offloading is given by $O(Z\lambda M\varepsilon_1\varepsilon_2 m_2)$.

Proof: See appendix A. □

Remark 1: Both the computational complexity of the RL based mobile offloading and that of the deep RL based mobile offloading schemes linearly increase with the number of the edge devices M. The computational complexity of the RL based mobile offloading scheme also depends on the total number of learning samples k, the number of the feasible transmit power levels Ω, and the number of the divided subtasks X. On the other hand, the computational complexity of the deep RL based mobile offloading scheme also increases with the size of the sampled mobile offloading experience Z, the length of the state sequence λ, the number of filters ε_1 and ε_2 in the critic network, and the m_2 kernels in the critic network. In this case, the deep learning parameters in Algorithm 2 are chosen as a tradeoff between the

Algorithm 2 Deep RL based mobile offloading

Initialize λ, $\alpha, \delta, \kappa, \rho, t$, β, Z, and $\mathcal{N}$
for $k = 1, 2, \cdots$ **do**
  Divide the tasks into X subtasks
  Evaluate $[\xi_{i,j}]_{1\le i\le M, 1\le j\le X}$
  Obtain $[b_i]_{1\le i\le M}$
  Formulate the state via (1)
  if $k \le \lambda$ **then**
    Randomly select $\boldsymbol{a}^{(k)} \in \mathbf{A}$
  else
    Input $\left\{\boldsymbol{s}^{(\chi)}, \boldsymbol{a}^{(\chi)}, \boldsymbol{s}^{(k)}\right\}_{k-\lambda\le\chi\le k-1}$ to the actor network
    Obtain $\boldsymbol{a}^{(k)}$ based on $\mathcal{N}$ and the actor network output $\boldsymbol{\omega}$
    Offload $a_2 X$ subtasks to edge device a_1 with transmit power a_3
    Process $(1-a_2)X$ subtasks locally
    Measure t and β
    Obtain the SINR ρ from edge device a_1
    Evaluate u via (3)
    Save $\boldsymbol{s}^{(k)}, \boldsymbol{a}^{(k)}$ and u to the replay buffer
  end if
  if $k > Z$ **then**
    Randomly sample Z experiences as $\boldsymbol{F}_{Z\times 4}$ from the replay buffer
    Update $\boldsymbol{\phi}$ with Adam [30]
    Update $\boldsymbol{\theta}$ via (4)
    $\boldsymbol{\phi}' = \kappa\boldsymbol{\phi} + (1-\kappa)\boldsymbol{\phi}'$
    $\boldsymbol{\theta}' = \kappa\boldsymbol{\theta} + (1-\kappa)\boldsymbol{\theta}'$
  end if
end for

computational complexity and the computational performance in mobile offloading process.

The mobile offloading process against the heavy interference and jamming attacks can be viewed as a dynamic mobile offloading game. In this game, the mobile device chooses edge device $a_1^{(k)} \in \{1, 2, \cdots, M\}$, offloading rate $a_2^{(k)} \in [0, 1]$, and transmit power $a_3^{(k)} \in [0, P_{\mathrm{T}}]$, while the jammer decides its jamming power $z^{(k)}$, with the resulting interference power received by edge device i given by $y_i^{(k)} \in [0, P_{\mathrm{J},i}^{\max}]$.

The edge devices are assumed to have cached the computational results with $\mu a_2 C$ bits of the $a_2 X$ subtasks in the previous offloading, with $\mu \in [0, 1]$ similar to [16]. According to [16], [34] and [35], the task computational latency and the energy consumption are assumed to be

$$t = \max\left\{\frac{a_2}{b_{a_1}} + \frac{\mu a_2}{\hat{b}_{a_1}},\ \frac{1-a_2}{l}\right\} C \tag{5}$$

$$\beta = (1-a_2)\,\varsigma C l^2 D^3 + \frac{a_2 a_3 C}{b_{a_1}}. \tag{6}$$

By (3), (5) and (6), the utility depends on the sharing gain, the computational latency, the energy consumption and the SINR of the offloading signals given by

$$u = a_2 C - w_{\mathrm{L}} \max\left\{\frac{a_2}{b_{a_1}} + \frac{\mu a_2}{\hat{b}_{a_1}},\ \frac{1-a_2}{l}\right\} C - w_{\mathrm{E}}\,(1-a_2)\,\varsigma C l^2 D^3 - \frac{w_{\mathrm{E}} a_2 a_3 C}{b_{a_1}} + \frac{w_{\mathrm{S}} a_3 h_{a_1}}{\sigma^2 + y_{a_1}}. \tag{7}$$

In the mobile offloading game, the utility of the jammer is assumed to be $u_{\mathrm{J}} = -u - \upsilon y_{a_1}$.

Theorem 2: The performance bound of the RL based mobile offloading in the offloading game under continuous offloading policy space is given by

$$t \ge \frac{C}{b_{i^*}} + \frac{\mu C}{\hat{b}_{i^*}} \tag{8}$$

$$\beta \ge \frac{P_{\mathrm{T}} C}{b_{i^*}} \tag{9}$$

$$u \le C - \frac{w_{\mathrm{L}} + w_{\mathrm{E}} P_{\mathrm{T}}}{b_{i^*}} C - \frac{w_{\mathrm{L}} \mu C}{\hat{b}_{i^*}} + \frac{w_{\mathrm{S}} P_{\mathrm{T}} h_{i^*}}{\sigma^2 + P_{\mathrm{J},i^*}^{\max}} \tag{10}$$

if (12) holds, where

$$i^* = \arg\max_{1\le i\le M}\left\{-\tfrac{w_{\mathrm{L}}+w_{\mathrm{E}}P_{\mathrm{T}}}{b_i} C - \tfrac{w_{\mathrm{L}}\mu C}{\hat{b}_i} + \tfrac{w_{\mathrm{S}} P_{\mathrm{T}} h_i}{\sigma^2 + P_{\mathrm{J},i}^{\max}}\right\}. \tag{11}$$

Proof: See appendix B. □

Remark 2: As shown in (12), at the bottom of the next page, the mobile device has to process a large number of the computational tasks and the channel gain to edge device i^* is greater than a threshold that relies on the maximum transmit power, the interference power and jamming power. The mobile device offloads all the computational task to edge device i^* with the maximum transmit power.

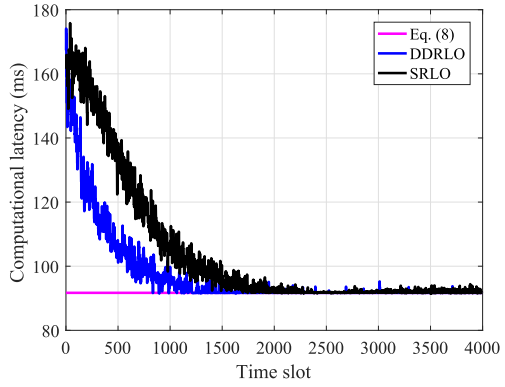
Corollary 1: SRLO can converge to the performance bound given by (8)-(10) after sufficient interactions, if

$$\begin{aligned}&\max_{\substack{1\le i\le M\\ 0\le n\le X\\ 0\le j\le\Omega}}\left\{\frac{n}{X} - w_{\mathrm{L}}\max\left\{\frac{n}{X b_i} + \frac{\mu n}{X\hat{b}_i},\ \frac{1}{l} - \frac{n}{Xl}\right\} - w_{\mathrm{E}}\varsigma l^2 D^3\right.\\ &\left. + \frac{w_{\mathrm{E}} n \varsigma l^2 D^3}{X} - \frac{w_{\mathrm{E}} n j P_{\mathrm{T}}}{X\Omega b_i} + \frac{w_{\mathrm{S}} j P_{\mathrm{T}} h_i}{C\Omega\left(\sigma^2 + P_{\mathrm{J},i^*}^{\max}\right)}\right\}\\ &\le 1 - \frac{w_{\mathrm{L}} + w_{\mathrm{E}} P_{\mathrm{T}}}{b_{i^*}} - \frac{w_{\mathrm{L}}\mu}{\hat{b}_{i^*}} + \frac{w_{\mathrm{S}} P_{\mathrm{T}} h_{i^*}}{C\left(\sigma^2 + P_{\mathrm{J},i^*}^{\max}\right)}\end{aligned} \tag{13}$$

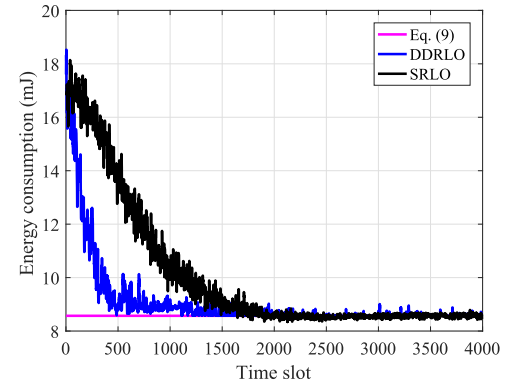
$$\frac{w_{\mathrm{S}} P_{\mathrm{T}} h_{i^*}}{\sigma^2 + P_{\mathrm{J},i^*}^{\max}} + \upsilon P_{\mathrm{J},i^*}^{\max} \le \min_{0\le j\le L}\left\{\frac{w_{\mathrm{S}} L P_{\mathrm{T}} h_{i^*}}{L\sigma^2 + j P_{\mathrm{J},i^*}^{\max}} + \frac{\upsilon j P_{\mathrm{J},i^*}^{\max}}{L}\right\}. \tag{14}$$

Proof: See appendix C. □

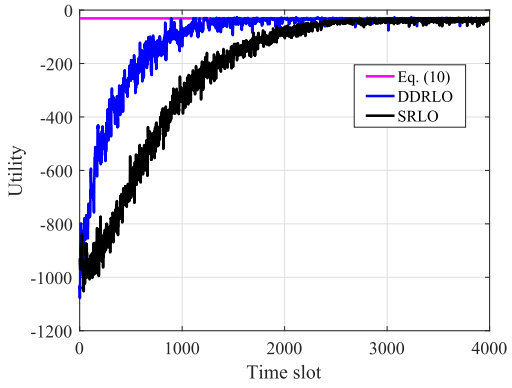
Simulation results in Fig. 3 show that SRLO and DDRLO converge to the bound given by (8)-(10) after 2800 time slots (about 3.1 seconds) and 1300 time slots (about 1.4 seconds), respectively. In addition, the convergence bound is very tight. The computational latency of SRLO after convergence is 91.8 ms, with less than 1.2‰ difference compared with the bound in (8). The energy consumption of SRLO reaches 8.59 mJ after 2800 time slots, and the difference from the bound in (9) is less than 2.3‰.



(a) Computational latency



(b) Energy consumption



(c) Utility

Fig. 3. Performance of the mobile offloading with 300 Kb tasks per time slot, 100 mW maximum power, 0.5 channel gain and 3.5 MHz radio bandwidth to the edge device against a jammer with 55 mW maximum jamming power and 0.4 unit jamming cost.

VII. Simulation Results

Simulations were performed for a MEC network as shown in Fig. 4 to evaluate the performance of the proposed RL based mobile offloading schemes. The three edge devices have time varying radio bandwidth with the mobile device ranging

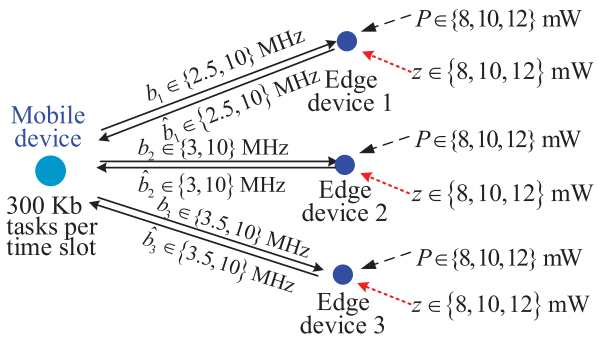


Fig. 4. Simulation setting for a mobile device that has to process 300 Kb computational tasks per time slot, in which three edge devices have dynamic bandwidth from 2.5 to 10 MHz. The jammer applies Q-learning to select the jamming power between 45 and 55 mW and the interference power randomly changes between 8 and 12 mW.

from 2.5 to 10 MHz by [1]. The mobile device uses the computational tasks generation model as presented in [36] to generate 300 Kb computational tasks per time slot such as the MMORPG game PlaneShift [37]. The mobile device locally processes tasks at 1 Mbps speed, uses 1000 CPU cycles to process each one-bit task, and transmits the others to the chosen edge device with power ranging from 0 to 100 mW. The interference power randomly changes over time between 8 and 12 mW. The effective capacitance coefficient of the mobile device $\varsigma = 10^{-28}$ according to [38]. The content popularity of the task follows the Zipf demand distribution according to [13]. The jammer applies Q-learning to select its jamming power between 45 and 55 mW with $\upsilon = 0.4$ to degrade the utility of the mobile device with reduced jamming energy consumption.

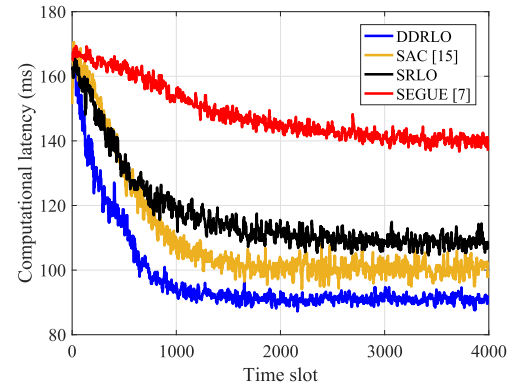
In the simulations, we set $\lambda = 3$, $\delta = 0.7$, and $\alpha = 0.3$. The actor network and critic network parameter setting is given by Table III to obtain high computational efficiency based on the simulation results not shown here. More specifically, both the actor network and the critic network have 2 Conv. layers and 4 FC layers, and include 16 filters with size 5×1 in Conv.1 and 32 filters with size 3×1 in Conv. 2. FCs 1-4 in the actor network have 200, 150, 150 and 3 neurons, respectively. The critic network has 200, 150 and 150 neurons in FCs 1-3, respectively, and one neuron in the output layer FC 4.

As shown in Fig. 5, the proposed RL based mobile offloading scheme exceeds the benchmark SEGUE in [7] with shorter computational latency, less energy consumption and higher utility, due to the risk level based on the computational latency in the offloading policy selection learning process. For example, this scheme saves energy consumption of the mobile device by 25.9%, reduces the computational latency of the tasks by 23.4% and increases the utility by 46.5% after 1500 time slots. The proposed DDRLO further improves the performance, i.e., DDRLO saves energy consumption

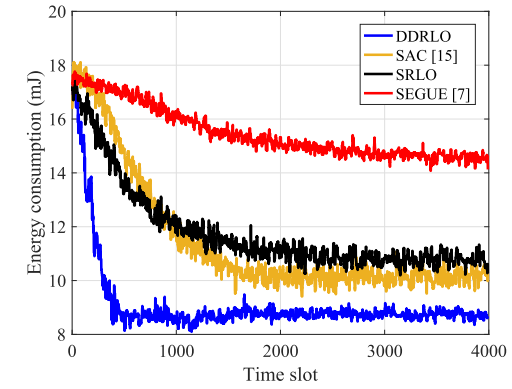
$$\upsilon \frac{\left(\sigma^2 + P_{\mathrm{J},i^*}^{\max}\right)^2}{w_{\mathrm{S}} P_{\mathrm{T}}} \leq h_{i^*} \leq \frac{C\left(\sigma^2 + P_{\mathrm{J},i^*}^{\max}\right)\left(w_{\mathrm{L}} b_{i^*}^{-1} + w_{\mathrm{L}} \mu \hat{b}_{i^*}^{-1} + w_{\mathrm{E}} P_{\mathrm{T}} b_{i^*}^{-1} - w_{\mathrm{L}} l^{-1} - w_{\mathrm{E}} \varsigma l^2 D^3 - 1\right)}{w_{\mathrm{S}} P_{\mathrm{T}}} \tag{12}$$

TABLE III
DEEP LEARNING PARAMETERS IN ALGORITHM 2

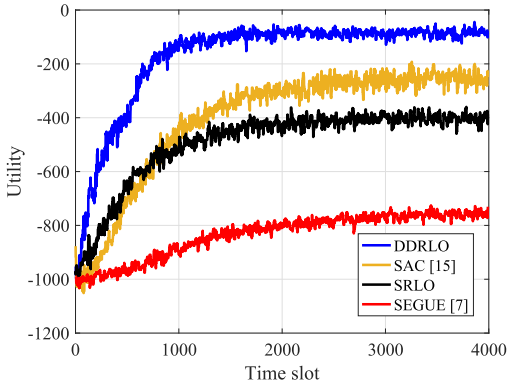
Network	Actor network				Critic network			
Layer	**Conv. 1**	**Conv. 2**	**FC 1**	**FC 4**	**Conv. 1**	**Conv. 2**	**FC 1**	**FC 4**
Input size	$1 \times 93 \times 1$	$16 \times 89 \times 1$	2784	150	$1 \times 96 \times 1$	$16 \times 92 \times 1$	2880	150
Number of the filters	16	32	200	3	16	32	200	1
Output size	$16 \times 89 \times 1$	$32 \times 87 \times 1$	200	3	$16 \times 92 \times 1$	$32 \times 90 \times 1$	200	1



(a) Computational latency

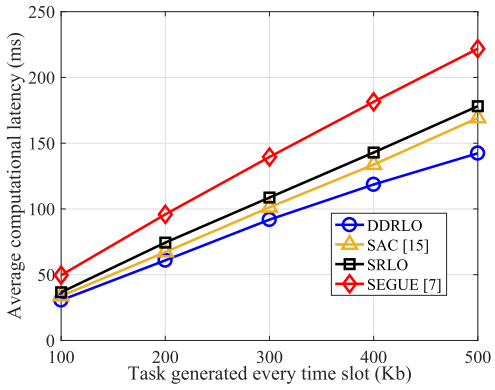


(b) Energy consumption

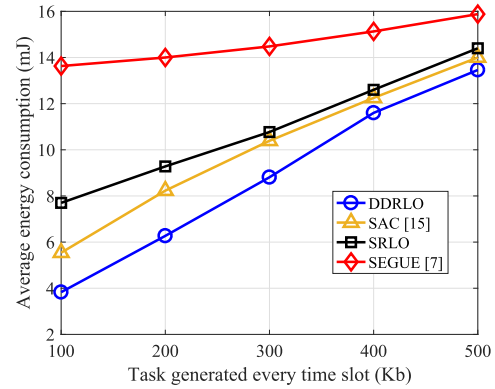


(c) Utility

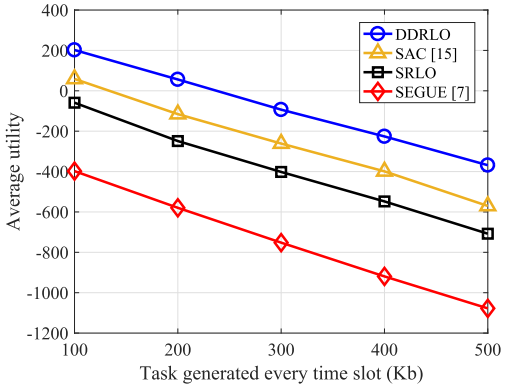
Fig. 5. Performance of the RL based mobile offloading with three edge devices each having random radio bandwidth from 2.5 to 10 MHz for the 300 Kb tasks in each time slot against smart jamming and random interference.



(a) Average computational latency



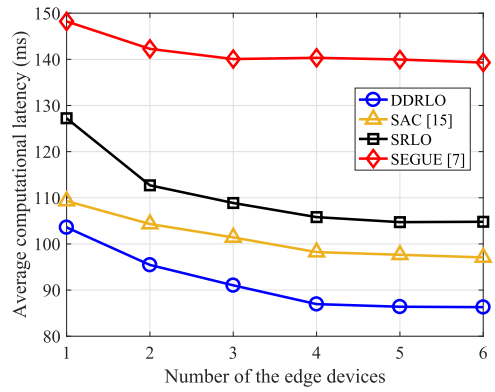
(b) Average energy consumption



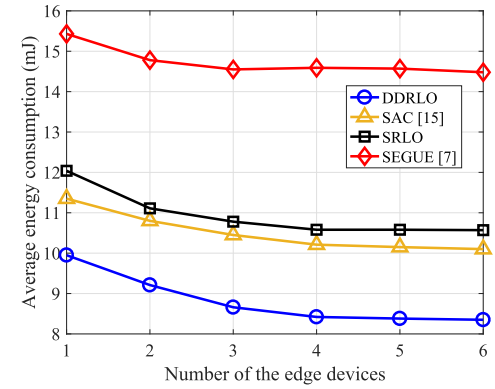
(c) Average utility

Fig. 6. Performance of the RL based mobile offloading averaged over 500 time slots with three edge devices each having random radio bandwidth from 2.5 to 10 MHz for the $100 \sim 500$ Kb computational tasks against smart jamming and random interference.

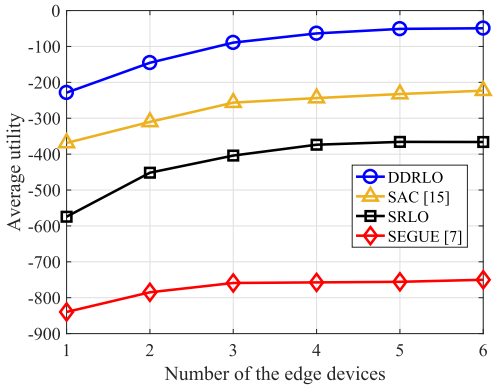
by 26.1%, reduces the computational latency by 18.4%, and improves the utility by 75.3%. DDRLO converges to the optimal offloading policy after 1500 time slots, which is 57.1% faster than SEGUE. The performance gain results from the actor-critic mechanism that effectively compresses the action and state spaces.



(a) Average computational latency



(b) Average energy consumption



(c) Average utility

Fig. 7. Performance of the RL based mobile offloading averaged over 500 time slots with 300 Kb tasks per time slot for six edge devices, with each having random radio bandwidth from 2.5 to 10 MHz against smart jamming and random interference.

Our proposed DDRLO reduces the computational latency of the tasks, saves energy consumption of the mobile device, and increases the utility compared with SAC in [15]. For example, this scheme reduces the computational latency by 14.0%, saves energy consumption by 27.9%, increases the utility by 22.0% at the 1000-th time slot, and saves the exploration time by 40%, as shown in Fig. 5. That is because this scheme uses the state sequence instead of the current state as the actor network input to extract the temporal correlations between the states and thus improve the computational performance. DDRLO decreases the computational latency to 92.6 ms after 1500 time slots, which is 22.8% less than the 120 ms best response time of the MMORPG game PlaneShift in [37]. The energy consumption converges to 8.7 mJ and satisfies the energy constraints of Nokia N810 and N900 devices in [38].

The computational performance averaged over 500 time slots with three edge devices for the $100 \sim 500$ Kb computational tasks is shown in Fig. 6. The computational latency increases from 30.6 to 142.3 ms and the energy consumption changes from 3.8 to 13.5 mJ as the task size changes from 100 to 500 Kb. As shown in Fig. 6, the proposed SRLO exceeds SEGUE, e.g., SRLO reduces the computational latency of the tasks by 21.2%, saves energy consumption of the mobile device by 16.7% and increases the utility by 40.9% with 400 Kb tasks per time slot. DDRLO further reduces the computational latency by 17.0%, saves energy consumption by 7.9% and improves the utility by 58.4%.

As shown in Fig. 7, the computational performance averaged over 500 time slots with 300 Kb tasks per time slot is provided for the six edge devices. The computational latency decreases by 16.1% to 87.0 ms as the number of the edge devices increases from 1 to 4 and decreases slightly afterwards, which is 13.0% faster than the 100 ms requirement of the shooter game Call-of-Duty 13 in [37]. The energy consumption decreases by 15.4% to 8.4 mJ as the number of edge devices increases from 1 to 4 and decreases slightly afterwards, which is 39.4% less than the 13.9 mJ energy consumption for a typical Android smartphone to process face recognition applications in [39]. Compared with SEGUE, SRLO reduces the computational latency by 25.2%, saves energy consumption by 27.4% and increases the utility by 51.6% with 5 edge devices. DDRLO further improves the computational performance, e.g., DDRLO reduces the computational latency by 17.5%, saves energy consumption by 20.8% and increases the utility by 86.9%.

VIII. Conclusion

In this paper, we have designed an RL based mobile offloading scheme for MEC to choose the edge device, the transmit power and the offloading rate to address smart jamming and heavy interference, and proposed a deep RL version that combines deep Q-network and the actor-critic algorithm to further improve the computational performance for the mobile devices that support deep learning. We have provided the computational complexity of the proposed scheme and its performance bound containing the computational latency of the tasks, the energy consumption and the utility of the mobile device based on game theory. Simulations have been performed for a mobile device that is connected with three edge devices with dynamic radio bandwidth, showing that this scheme exceeds SEGUE in [7] with shorter computational latency, less energy consumption and higher utility. For example, the deep RL based mobile offloading scheme saves energy consumption by 42.3%, reduces the computational latency by 37.0% and improves the utility by 89.1% after 2000 time slots against smart Q-learning based jamming and heavy interference.

In the future, an interesting direction in MEC is to apply federated reinforcement learning to the avoid privacy leakage in the mobile offloading and reduce the communication overhead of the mobile devices. In addition, the proposed mobile offloading scheme can use the model planning method to incorporate the known edge computing information, such as the distances between the mobile device and the edge devices to accelerate the learning speed against jamming in a dynamic MEC network.

APPENDIX A
PROOF OF THEOREM 1

Proof: According to [32], the computational complexity of the RL based mobile offloading scheme is given by $O(kMX\Omega)$.

According to the critic network architecture and the input size, we have

$$m_1\left(7\lambda+\lambda M+M-m_1+8\right) \ll \varepsilon_2 m_2\left(7\lambda+\lambda M+M-m_1-m_2+9\right). \quad (15)$$

Similar to [33], the computational complexity of the deep RL based mobile offloading scheme is given by

$$\begin{aligned} &O\Big(Z\varepsilon_1\big(m_1\left(7\lambda+\lambda M+M-m_1+8\right) \\ &\quad +\varepsilon_2 m_2\left(7\lambda+\lambda M+M-m_1-m_2+9\right)\big)\Big) \\ &= O\big(Z\varepsilon_1\varepsilon_2 m_2\left(7\lambda+\lambda M+M-m_1-m_2+9\right)\big) \quad (16) \\ &= O\big(Z\lambda\varepsilon_1\varepsilon_2 m_2\left(7+M\right)\big) \quad (17) \\ &= O\left(Z\lambda M\varepsilon_1\varepsilon_2 m_2\right), \quad (18) \end{aligned}$$

where (16) is obtained if (15) holds, and (17) assumes $7\lambda + M\lambda \gg M - m_1 - m_2 + 9$. □

APPENDIX B
PROOF OF THEOREM 2

Proof: By (7), $u_{\mathrm{J}} = -u - \upsilon y_{a_1}$ and i^* given by (11), if (12) holds, we have

$$\frac{\partial u_{\mathrm{J}}\left([i^*,1,P_{\mathrm{T}}],y_{i^*}\right)}{\partial y_{i^*}} = \frac{w_{\mathrm{S}}P_{\mathrm{T}}h_{i^*}}{(\sigma^2+y_{i^*})^2}-\upsilon \geq 0, \quad \forall y_{i^*}\in\left[0,P_{\mathrm{J},i^*}^{\max}\right]. \quad (19)$$

Thus, by (3), we have

$$u_{\mathrm{J}}\left([i^*,1,P_{\mathrm{T}}],P_{\mathrm{J},i^*}^{\max}\right) \geq u_{\mathrm{J}}\left([i^*,1,P_{\mathrm{T}}],y_{i^*}\right), \quad \forall y_{i^*}\in\left[0,P_{\mathrm{J},i^*}^{\max}\right]. \quad (20)$$

By (7), we have

$$\frac{\partial^2\ u\left(\boldsymbol{a},P_{\mathrm{J},i^*}^{\max}\right)}{\partial a_2^2}\frac{\partial^2\ u\left(\boldsymbol{a},P_{\mathrm{J},i^*}^{\max}\right)}{\partial a_3^2}-\left(\frac{\partial^2\ u\left(\boldsymbol{a},P_{\mathrm{J},i^*}^{\max}\right)}{\partial a_2 a_3}\right)^2 = -\frac{w_{\mathrm{E}}^2C^2}{b_{a_1}^2}<0, \forall 1\leq a_1\leq M, a_2\in[0,1], a_3\in[0,P_{\mathrm{T}}]. \quad (21)$$

According to [40], by (7), (21) and i^* given by (11), if (12) holds, we have

$$\begin{aligned} &u\Big(\left[i^*,1,P_{\mathrm{T}}\right],P_{\mathrm{J},i^*}^{\max}\Big) \\ &= C-\frac{w_{\mathrm{L}}+w_{\mathrm{E}}P_{\mathrm{T}}}{b_{i^*}}C-\frac{w_{\mathrm{L}}\mu C}{\hat{b}_{i^*}}+\frac{w_{\mathrm{S}}P_{\mathrm{T}}h_{i^*}}{\sigma^2+P_{\mathrm{J},i^*}^{\max}} \\ &\geq a_2C-w_{\mathrm{L}}\max\left\{\frac{a_2}{b_{a_1}}+\frac{\mu a_2}{\hat{b}_{a_1}},\ \frac{1-a_2}{l}\right\}C \\ &\quad -w_{\mathrm{E}}\left(1-a_2\right)\varsigma Cl^2D^3-\frac{w_{\mathrm{E}}a_2a_3\ C}{b_{a_1}}+\frac{w_{\mathrm{S}}a_3h_{a_1}}{\sigma^2+P_{\mathrm{J},i^*}^{\max}} \\ &= u\Big(\boldsymbol{a},P_{\mathrm{J},i^*}^{\max}\Big), \forall 1\leq a_1\leq M, a_2\in[0,1], a_3\in[0,P_{\mathrm{T}}]. \quad (22) \end{aligned}$$

Thus, by (20) and (22), we can prove that $\left([i^*,1,P_{\mathrm{T}}],P_{\mathrm{J},i^*}^{\max}\right)$ is a NE of the mobile offloading game under a continuous offloading policy space. Therefore, by (5)-(7), we have the performance bound given by (8)-(10). □

APPENDIX C
PROOF OF COROLLARY 1

Proof: By (7), if (13) and (14) hold, we have (20) and (22). Thus, $\left([i^*,1,P_{\mathrm{T}}],P_{\mathrm{J},i^*}^{\max}\right)$ is a NE of the mobile offloading game. By (22), we have $\boldsymbol{a}^*=[i^*,1,P_{\mathrm{T}}]$. According to [41], the tabular RL algorithm under the discrete action set $\mathbf{A}$ converges to $\boldsymbol{a}^*$ after sufficient interactions. Thus, if (13) and (14) hold, SRLO converges to the performance bound given by (8)-(10). □

REFERENCES

[1] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[2] Y. He, F. R. Yu, N. Zhao, V. C. M. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 31–37, Dec. 2017.

[3] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, "Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 64–69, May 2019.

[4] W. Liu, J. Cao, L. Yang, L. Xu, X. Qiu, and J. Li, "Appbooster: Boosting the performance of interactive mobile applications with computation offloading and parameter tuning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 6, pp. 1593–1606, Jun. 2017.

[5] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7432–7445, Aug. 2017.

[6] L. Xiao, X. Wan, C. Dai, X. Du, X. Chen, and M. Guizani, "Security in mobile edge caching with reinforcement learning," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 116–122, Jun. 2018.

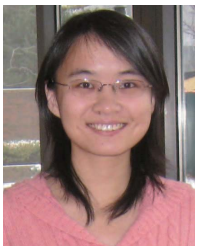
[7] W. Zhang, Y. Hu, Y. Zhang, and D. Raychaudhuri, "SEGUE: Quality of service aware edge cloud service migration," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Luxembourg City, Luxembourg, Dec. 2016, pp. 344–351.

[8] L. Xiao, Y. Li, X. Huang, and X. Du, "Cloud-based malware detection game for mobile devices with offloading," *IEEE Trans. Mobile Comput.*, vol. 16, no. 10, pp. 2742–2750, Oct. 2017.

[9] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10190–10203, Nov. 2018.

[10] Q. Cui *et al.*, "Stochastic online learning for mobile edge computing: Learning from changes," *IEEE Commun. Mag.*, vol. 57, no. 3, pp. 63–69, Mar. 2019.

[11] Q. Qi *et al.*, "Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4192–4203, May 2019.
[12] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.
[13] A. S. Cacciapuoti, M. Caleffi, M. Ji, J. Llorca, and A. M. Tulino, "Speeding up future video distribution via channel-aware caching-aided coded multicast," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 8, pp. 2207–2218, Aug. 2016.
[14] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Commun. Mag.*, vol. 51, no. 4, pp. 142–149, Apr. 2013.
[15] X. Qiu, L. Liu, W. Chen, Z. Hong, and Z. Zheng, "Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8050–8062, Aug. 2019.
[16] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.
[17] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.
[18] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4804–4814, Jun. 2019.
[19] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Hong Kong, Apr. 2015, pp. 1894–1902.
[20] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.
[21] 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.
[22] Z. Zhou, P. Liu, Z. Chang, C. Xu, and Y. Zhang, "Energy-efficient workload offloading and power control in vehicular edge computing," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, Barcelona, Spain, Apr. 2018, pp. 191–196.
[23] C.-F. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4132–4150, Jun. 2019.
[24] S. Li *et al.*, "Joint admission control and resource allocation in edge computing for Internet of Things," *IEEE Netw.*, vol. 32, no. 1, pp. 72–79, Jan. 2018.
[25] X. Chen, J. Wu, Y. Cai, H. Zhang, and T. Chen, "Energy-efficiency oriented traffic offloading in wireless networks: A brief survey and a learning approach for heterogeneous cellular networks," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 4, pp. 627–640, Apr. 2015.
[26] 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.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.
[27] F. Liu *et al.*, "Gearing resource-poor mobile devices with powerful clouds: Architectures, challenges, and applications," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 14–22, Jun. 2013.
[28] M. Chen, Y. Qian, Y. Hao, Y. Li, and J. Song, "Data-driven computing and caching in 5G networks: Architecture and delay analysis," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 70–75, Feb. 2018.
[29] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, San Diego, CA, USA, May 2015, pp. 1–15.
[30] X. Wan, G. Sheng, Y. Li, L. Xiao, and X. Du, "Reinforcement learning based mobile offloading for cloud-based malware detection," in *Proc. GLOBECOM-IEEE Global Commun. Conf.*, Singapore, Dec. 2017, pp. 1–6.
[31] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*. [Online]. Available: http://arxiv.org/abs/1509.02971
[32] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan, "Is Q-learning provably efficient?" in *Proc. Conf. Adv. Neural Inf. Process. Syst. (NIPS)*, Montreal, QC, Canada, Dec. 2018, pp. 4868–4878.
[33] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, Jun. 2015, pp. 5353–5360.
[34] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.
[35] 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.
[36] 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.
[37] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.
[38] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. USENIX Conf. Hot Topics Cloud Comput. (HotCloud)*, Boston, MA, USA, Jun. 2010, pp. 1–7.
[39] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," *IEEE Trans. Mobile Comput.*, vol. 18, no. 2, pp. 319–333, Feb. 2019.
[40] T. R. Oliveira, M. Krstic, and D. Tsubakino, "Extremum seeking for static maps with delays," *IEEE Trans. Autom. Control*, vol. 62, no. 4, pp. 1911–1926, Apr. 2017.
[41] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.



Liang Xiao (Senior Member, IEEE) received the B.S. degree in communication engineering from the Nanjing University of Posts and Telecommunications, China, in 2000, the M.S. degree in electrical engineering from Tsinghua University, China, in 2003, and the Ph.D. degree in electrical engineering from Rutgers University, NJ, in 2009.

She was a Visiting Professor with Princeton University, Virginia Tech, and the University of Maryland, College Park. She is currently a Professor with the Department of Information and Communication Engineering, Xiamen University, Xiamen, China. She was a recipient of the Best Paper Award for 2016 INFOCOM Big Security WS and 2017 ICC. She has served as an Associate Editor for IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY and a Guest Editor for IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING.

Xiaozhen Lu received the B.S. degree in communication engineering from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2017. She is currently pursuing the Ph.D. degree with the Department of Information and Communication Engineering, Xiamen University, Xiamen, China. Her research interests include network security and wireless communications.

Tangwei Xu (Student Member, IEEE) received the B.S. degree in electronic information engineering from the Nanjing University of Science and Technology, Nanjing, China, in 2018. He is currently pursuing the M.S. degree with the Department of Information and Communication Engineering, Xiamen University, Xiamen, China. His current research interests include network security and wireless communications.

Xiaoyue Wan received the B.S. and M.S. degrees in information and communication engineering from Xiamen University, Xiamen, China, in 2016 and 2019, respectively.

Wen Ji (Senior Member, IEEE) received the M.S. and Ph.D. degrees in communication and information systems from Northwestern Polytechnical University, China, in 2003 and 2006, respectively. From 2014 to 2015, and in 2018, she was a Visiting Scholar with the Department of Electrical Engineering, Princeton University, USA. She is currently a Professor with the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, and the Peng Cheng Laboratory, Shenzhen, China. Her research interests include multimedia communication and networking, multimedia economics, network economics, video coding, channel coding, information theory, optimization, and ubiquitous computing. She has 16 issued patents and more than 80 refereed publications in journals and conferences related to her areas of interest.

Yanyong Zhang (Fellow, IEEE) received the B.S. degree from the University of Science and Technology of China (USTC) in 1997 and the Ph.D. degree from Penn State University in 2002.

From 2002 to 2018, she was on the faculty of the Electrical and Computer Engineering Department, Rutgers University. She was also a member of the Wireless Information Networks Laboratory (Winlab). In July 2018, she joined the School of Computer Science and Technology, USTC. She has 21 years of research experience in the areas of sensor networks, ubiquitous computing, and high-performance computing, and has published more than 110 technical articles in these fields. She has received the NSF CAREER Award in 2006. She currently serves as an Associate Editor for several journals, including IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON SERVICES COMPUTING, and *Smart Health* (Elsevier).