Int. J. Electron. Commun. (AEÜ) 118 (2020) 153134

Contents lists available at ScienceDirect

International Journal of Electronics and Communications (AEÜ)

journal homepage: www.elsevier.com/locate/aeue

Regular paper

Novel method of mobile edge computation offloading based on evolutionary game strategy for IoT devices

Yuya Cui *, Degan Zhang, Ting Zhang, Lu Chen, Mingjie Piao, Haoli Zhu

Key Laboratory of Computer Vision and System, Tianjin University of Technology, Ministry of Education, Tianjin 300384, China
Tianjin Key Lab of Intelligent Computing & Novel Software Technology, Tianjin University of Technology, Tianjin 300384, China

ARTICLE INFO

Article history:
Received 17 December 2019
Accepted 16 February 2020

Keywords:
Internet of Things
Computation offloading
Mobile edge computing
Evolutionary game theory
Reinforcement learning

ABSTRACT

Due to the limited computing resources and energy of IoT devices, complex computing tasks are offloaded to sufficient computing servers, such as Cloud Center. However, offloading may increase the latency and congestion of the IoT network. Mobile Edge Computing (MEC) is a promising approach, which can decrease the delay and energy consumption of IoT devices significantly. In this paper, we investigate the problem of multi-user computation offloading under dynamic environment. Considering the channel interference when multiple IoT devices offload computing tasks via wireless channels at the same time, we formulate the computation offloading as an evolutionary game model. We use the replicator dynamics to analyze the evolutionary process of IoT devices and prove that multi-user computation offloading exists unique Evolutionary Stability Strategy (ESS). Finally, we design an evolutionary game algorithm based on reinforcement learning in practical application scenarios. Experiments can verify the convergence and performance of the proposed algorithm in multi-user scenarios.

1. Introduction

With the development of Internet of Things (IoT) and WSN [1–13], IoT devices are widely used in smart factories. However, the computing resources and energy of IoT devices are limited [14–31], so the growth of IoT devices has brought enormous challenges to IoT system. The traditional solution is to introduce cloud computing [32] and offload computing tasks to servers with sufficient computing resources. However, cloud services are centralized to process data. Multiple IoT devices offload data to the cloud center at the same time, which may lead to serious network congestion and increase latency and energy consumption of IoT devices, especially for some delay-sensitive tasks [33]. Mobile edge computing (MEC) [34–40] can effectively solve the above problems. In MEC system, the computing power of edge servers is much higher than that of IoT devices, and edge severs are deployed closer to the devices on the edge of network, so offloading computing tasks to edge servers can reduce latency and energy consumption. However, the computing resources of edge servers are much lower than those of cloud center, and IoT devices communicate with MEC through wireless channels. Therefore, It is necessary to design an effective offloading strategy to determine where to perform computing tasks(local or edge server). That can effectively reduce the congestion of wireless channels, so as to improve offloading efficiency of IoT devices and decrease the delay and energy consumption of devices. Therefore, the computation offloading in MEC has been extensively researched [41–45].

Evolutionary Game Theory (EGT) is a method of game theory, which is widely used in resource competition environments [46]. EGT is different from the traditional game method, which does not require the player to be completely rational. It selects the optimal strategy through continuous learning and trial and error. Different IoT devices have their own computing tasks and they do not understand others' strategies, therefore, EGT is very suitable for the multi-user computation offloading scenario. In this paper, IoT devices adjust strategies based on the dynamic environment. We analyze the evolution process of offloading strategy through replicator dynamics, and each IoT device has no incentive to change its strategy when it reaches ESS. The main contributions are as follows:

1) By considering the transmission delay and energy consumption, multi-user computation offloading problem in MEC system is considered as an evolutionary game model. IoT devices are considered as game players, and the computing way (locally or offloaded to edge servers via channels) is considered as the strategy space of the game.

* Corresponding author.
E-mail address: 844511468@qq.com (Y. Cui).

https://doi.org/10.1016/j.aeue.2020.153134
1434-8411/© 2020 Elsevier GmbH. All rights reserved.

2) We study the process of updating the offloading strategy by the replicator dynamics, and further prove that there is a unique ESS in replicator dynamics.
3) In practice, we design an evolutionary game algorithm based on reinforcement learning. Each IoT device chooses and updates strategies according to Q learning algorithm, and finally achieves ESS through continuous learning. The convergence of the proposed algorithm is proved.

The remainder of this paper is organized as follows: In the second section, we give the related works; the third section introduces the system model; the fourth section introduces the multi-user computation offloading evolutionary game model; the fifth section gives the realization of the evolutionary game model in the distributed environment; Section six gives experimental results and analysis; we summarize the article in Section seven and give future work.

2. Related works

Currently, researches on computation offloading mainly focus on the single-user scenario. By considering the local overhead and limited communication and computing resources, Yu S et al. [47] formulated the offloading decision as a multi-label classification problem, and used deep learning method to minimize the consumption of computation. Zhang Y et al. regarded the offloading process as a Markov process to obtain an optimal strategy to minimize the calculation and mobile users' energy consumption [48]. Kwak J et al. performed computation offloading to minimize energy consumption by considering three moving factors: cloud offloading strategy, computing task allocation and CPU processing speed [49]. Huang D et al. [50] proposed a dynamic offloading method based on Lyapunov optimization. Experimental results showed that the algorithm had lower algorithm complexity and better performance in energy consumption. Wang J et al. proposed a new adaptive off-line decision transmission scheduling scheme. Firstly, an adaptive offloading model was designed, and then they decreased the complexity of the algorithm by introducing Lyapunov optimization [51].

There are not many researches on multi-user offloading strategies. Yang L et al. effectively improved the bandwidth of wireless network by introducing genetic algorithm and increased the throughput of data transmission [52]. Zhao Y et al. designed a heuristic method with low complexity for efficient offloading when the delay requirement was satisfied [53]. Sardellitti et al. optimized wireless resources and computing resources jointly, to minimize the energy consumption of the total users while satisfying the delay [54].

All the literatures mentioned above are centralized processing of computation offloading without considering the distributed environment. Chen X et al. proposed a distributed computation offloading algorithm that can achieve a Nash equilibrium, derive the upper bound of convergence time, and quantify its price of anarchy [55]. Literature [56] analyzed the application of game theory and reinforcement learning in computing task offloading for MEC. Literature [57] modeled multi-user offloading as a non-cooperative game, and then proved that it was a potential game with at least one pure strategy Nash equilibrium (NEP). The proposed FDCO algorithm can maximize the number of beneficial mobile cloud computing devices. Zheng J C et al. modeled the mobile user's offloading decision as a stochastic game model by considering whether the mobile device was active in a dynamic environment [58], which proved that it was equivalent to a potential game with weights, and there was at least a Nash equilibrium. Thinh Q D et al. established a multi-user offloading model based on game theory and Q learning [59]. All the above literatures regard all mobile users as completely rational subjects, which does not conform to the actual application scenario, and whether the participants are completely rational has a great impact on the experimental results. In this paper, we regard the total consumption of IoT devices during computation offloading as a utility function to establish an evolutionary game model and prove the convergence under the replicator dynamics. Considering the practical application, we propose an evolutionary game based on reinforcement learning method for computation offloading.

3. System model

The system model is shown in Fig. 1. We consider that the system has N IoT devices, $N = \{1, 2, \ldots, n\}$. And each device i has a computationally intensive task T_i. Each IoT device can offload local computing tasks to edge servers via wireless access points (AP) and has the option of executing locally and offloading tasks to edge servers. Suppose there are M available wireless channels which can be denoted by $M = \{1, 2, \ldots, m\}$.$s_i = 0$ means that the computing task is executed locally, and $s_i > 0$ means that the computing task is performed at the edge server, that is, $s_i \in \{0\} \cup M$ is used to indicate the offloading policy of device i. When device i selects different channels for offloading, its offloading rate is different.

3.1. Communication model

The channels between IoT devices and AP are time-varying and follow Rayleigh decay. $g_{i,p} = (d_{i,p})^{-\alpha}\beta_{i,p}$ represents the instantaneous channel gain between device i and AP, where$d_{i,p}$ represents the distance between device i and AP, α is the path loss index, and $\beta_{i,p}$ is the Rayleigh fading factor.

Because of the time variability of the channel, $\beta_{i,p}$ also changes with time. When device i chooses to offload the computing task to the edge server, that is, $s_i > 0$, we can get the upload rate [55] of device i as follows:

$$R_i(s) = B\log_2\left(1 + \frac{p_i g_{i,p}}{\sigma_0 + \sum\limits_{j \in s_{-i}: s_j = s_i} p_j g_{j,p}}\right) \tag{1}$$

where B represents the channel bandwidth, p_i represents the transmission power of device i, $g_{i,p}$ represents the channel gain from device i to AP, s_{-i} represents all devices' strategies except device i, and σ_0 represents the background noise. From Eq. (1), we can see that when multiple devices simultaneously offload computing tasks to the edge server, channel interference occurs, which will decrease the offloading efficiency and increase the delay and energy consumption.

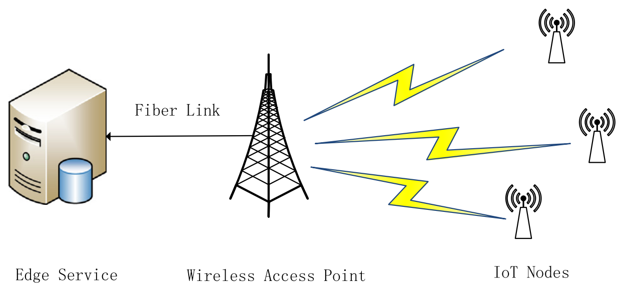


Fig. 1. System model.

3.2. Computation model

In this paper, each IoT device has an intensive task $T_i = \left(b_i, d_i^{loc}, d_i^{rem}\right)$, where b_i represents the size of date mainly including system settings, input parameters, and program code. d_i^{loc} represents the CPU resources of executing task T_i locally. d_i^{rem} represents the CPU resource of executing task T_i on the edge server. We can get the value of b_i, d_i^{loc}, d_i^{rem} from [60,61]. We believe that when the IoT device performs a computing task, either the task is executed locally or the task is executed on the edge server, and there is no case where local and edge servers execute the same task. Next, we describe the consumption of executing the computing task locally and on the edge server.

1) Local Computing: F_i^{loc} represents the computation ability of IoT device i (i.e., CPU cycles per second). Each IoT device has different computation ability. T_i indicates the executing time when the computing task is executed locally:

$$t_i^{loc} = \frac{d_i^{loc}}{F_i^{loc}} \tag{2}$$

The energy consumed is:

$$E_i^{loc} = \gamma_i d_i^{loc} \tag{3}$$

where γ_i represents the coefficient of energy consumption per CPU cycle, which can be obtained according to the method [62]. We calculate the time and energy consumption as the total overhead of device i. From formulas (2) and (3), we can get that the total overhead is O_i^{loc}:

$$O_i^{loc} = \lambda_i^t t_i^{loc} + \lambda_i^e E_i^{loc} \tag{4}$$

where λ_i^t and λ_i^e represent the calculation time weight and energy consumption weight of the device i, $\lambda_i^t, \lambda_i^e \in [0,1]$. When the energy of the device is low, let $\lambda_i^e = 1$, $\lambda_i^t = 0$, that is, the strategy of low energy consumption is given priority in the offloading decision. When time-sensitive tasks are performed, let $\lambda_i^e = 0, \lambda_i^t = 1$, that is, the strategy with less executing time is given priority in the offloading decision. If the device is not in two extreme conditions, the value of λ_i^e, λ_i^t can be set flexibly, taking into account both time and energy consumption.

2) Edge Server Computing: IoT devices can offload task T_i to edge server via wireless channels. In this paper, we ignore the time that the server returns the result of the calculation, because the return is much smaller than the input data. Therefore, when the computing task is offloaded to the edge server, there are three main consumption aspects: (1) time of transferring input data to the server; (2) energy consumption during data transmission; (3) execution time at the server. According to formula (1), we can get the transmission time $t_i^{rem}(s)$ and the transmission energy consumption $E_i^{rem}(s)$:

$t_i^{rem}(s) = \frac{b_i}{R_i(s)}$ and $E_i^{rem}(s) = \frac{p_i b_i}{R_i(s)}$

The time of executing task on the edge server is $t_i^{server} = \frac{d_i^{rem}}{F_i^{server}}$. Where F_i^{server} represents the CPU resources that the server assigns to the device. From the above three aspects, we can conclude that the total consumption of offloading computing tasks to the edge server isO_i^{rem}:

$$\begin{aligned} O_i^{rem}(s) &= \lambda_i^t\left(t_i^{rem}(s) + t_i^{server}\right) + \lambda_i^e E_i^{rem}(s) \\ &= \lambda_i^t\left(\frac{b_i}{R_i(s)} + \frac{d_i^{rem}}{F_i^{server}}\right) + \lambda_i^e \frac{p_i b_i}{R_i(s)} \\ &= \frac{b_i\left(\lambda_i^t + \lambda_i^e p_i\right)}{R_i(s)} + \lambda_i^t t_i^{server} \end{aligned} \tag{5}$$

According to the system model proposed in this section, we will design an evolutionary game model for multi-user computation offloading problem.

4. Multi-user evolutionary game strategy

In this section, we make full use of evolutionary game theory to solve the resource competition problem in multi-user computation offloading. Firstly, we establish an evolutionary game model for multi-user computation offloading in a dynamic environment. Secondly, we use replicator dynamics to analyze the evolution of IoT devices. Finally, it is proved that the evolutionary game model has a unique evolutionary stability strategy (ESS).

4.1. Evolutionary game model

For the multi-user computation offloading evolutionary game, participants are N IoT devices. They form two populations in the game, that is, they can perform computing tasks locally, or they can offload computing tasks to the edge server. In the dynamic environment, device i does not understand the strategies adopted by other participants and their payments. Adjusting their own strategies through continuous evolution and trial and error to maximize the fitness value. The goal of this paper is to minimize the energy consumption, delay and maximize the fitness value (utility). The number of devices that choose the pure strategy $s_k, k \in \{0, 1, 2 \ldots, m\}$ at time t is expressed as $n_k(t)$, $n_k(t) \geqslant 0$, and the logarithmic utility function [63] is used to represent the revenue $B_i(s_k)$ of the device i selecting strategy s_k, which is expressed as follows:

$$B_i(s_k) = \frac{1}{\ln\left(\frac{1+\beta}{\beta}\right)}[\ln(s_k + \beta) - \ln\beta] \tag{6}$$

where β represents utility constant. $P_i(s_k)$ represents the payment of device i:

$$P_i(s_k) = \begin{cases} O_i^{rem}(s_k) & s_k > 0 \\ O_i^{loc} & s_k = 0 \end{cases} \tag{7}$$

At time t, when $s_k > 0$, device i performs the computing task at the edge server, at this time the payment function of i is $O_i^{rem}(s_k)$. When $s_k = 0$, device i performs the computing task locally, and the payment function of i is O_i^{loc}. Therefore, we can get the utility function $u_k(t)$ of device i selecting strategy s_k in the population:

$$u_k(t) = \begin{cases} \sum_{i=1}^{n_k(t)} B_i(s_k) - P_i(s_k)/n_k(t) & n_k(t) > 0 \\ 0 & n_k(t) = 0 \end{cases} \tag{8}$$

4.2. Replicator dynamics

Replicator dynamics treats individuals who choose a kind of strategy as a whole to study, reducing the process of deducing individual behavior. In the replicator dynamics model, when a strategy can generate a higher benefit than the group average expectation, the proportion of the population using this strategy increases, while the proportion of the population using this strategy decreases. We use the replicator dynamics equation to simulate the distribution trajectory of the strategy in the population. The total number of all players in the population is $\sum_k n_k(t) = N$. The ratio of the number of selecting strategy k to the entire population is $x_k(t)$. Vector $X(t) = [x_1(t), \ldots, x_k(t), \ldots, x_m(t)]$ represents the distribution of the selected strategy of the device in population, where $x_k(t) \in [0,1]$ and$\sum_k x_k(t) = 1, t \geqslant 0$. Therefore, the expected utility $\pi_k(t)$ of the device i selecting strategy k at time t can be calculated as follows:

$$\begin{aligned}\pi_k(t) &= \sum_{n_k(t)=1}^{N} C_N^{n_k(t)} (x_k(t))^{n_k(t)} (1-x_k(t))^{N-n_k(t)} u_k(t)\\ &= \frac{\sum_{i=1}^{n_k(t)} B_i(s_k) - P_i(s_k)}{x_k \cdot N} \sum_{n_k(t)=1}^{N} \frac{N!}{n_k!(N-n_k)!} (x_k)^{n_k} (1-x_k)^{N-n_k}\\ &= \frac{W_k}{x_k \cdot N}\end{aligned} \tag{9}$$

To simplify, let $W_k = \sum_{i=1}^{n_k(t)} B_i(s_k) - P_i(s_k)$. The average utility $\bar{\pi}(x)$ of a population can be expressed as:

$$\begin{aligned}\bar{\pi}(x) &= \sum_{k=1}^{m} x_k(t)\pi_k(t)\\ &= \frac{\sum_{k=1}^{m} W_k}{N}\end{aligned} \tag{10}$$

From Eqs. (9) and (10), the replicator dynamics can be obtained as follows:

$$\begin{aligned}\dot{x}_k(t) &= \gamma x_k(t)\left[\pi_k(t) - \bar{\pi}(x)\right]\\ &= \gamma \frac{W_k}{N} - \gamma x_k \sum_{j=1}^{m} \frac{W_j}{N}\end{aligned} \tag{11}$$

where γ is a normal number, indicating the learning efficiency of the population. $\pi_k(t)$ represents the utility of selecting k strategy population at time t. $\dot{x}_k(t)$ represents the rate of change of $x_k(t)$. In the replicator dynamics model, the strategy with higher fitness value can reproduce more generations. In this paper, the higher the utility value of the IoT device, the higher the fitness value. Therefore, when the utility value of the k strategy is higher than the average utility, the number of IoT devices that select the k strategy will increase in the later evolutionary learning process.

4.3. Analysis of evolutionary stability strategy

Evolutionary stability strategy (ESS) is a standard to evaluate whether an evolutionary game is stable or not. When game achieves evolutionary stability strategy, all participants in the population adopt the same strategy. In the case of natural selection, no mutant gene can invade the population [64]. We use evolutionary stability strategies to analyze the evolutionary equilibrium and stability of multi-user computation offloading.

Theorem 1. *The replicator dynamics Eq. (11) has a globally unique solution* $X^* = [x_1^*, \ldots, x_k^*, \ldots x_m^*]$.

Proof. We use Lipschitz continuity to prove that Eq. (11) has a unique solution. Lipschitz continuity can be simply defined as: function $F(x)$ is defined on a certain interval, and there is a constant $L > 0$, so that for any x, y on the interval satisfies $|F(x) - F(y)| < L|x - y|$, $F(x)$ satisfies the Lipschitz condition, and $F(x)$ has a unique solution. Firstly, we convert $\dot{x}_k(t)$ in formula (11) into $F_k(t, X)$, where X represents the strategy distribution vector of the population. Given an initial condition $X(0) = [x_1(0), \ldots, x_k(0), \ldots, x_m(0)] \in \Omega$, where Ω is state space containing all possible population distribution states.$D = \{(t, x_1, \ldots, x_k, \ldots x_m) : \{|t| \leqslant a, |x_k - x_k(0)| \leqslant b\}\}$ is a continuous interval of $F_k(t, X)$. Let$L = \max\left\{\gamma \sum_{j=1}^{m} \frac{W_j}{N}\right\}$, given any two values $(t, x_k), (t, y_k)$ of the Dinterval, we can get:

$$\begin{aligned}|F_k(t,x_k) - F_k(t,y_k)| &= \gamma\left|\left(\frac{W_k}{N} - x_k\sum_{j=1}^{m}\frac{W_j}{N}\right) - \left(\frac{W_k}{N} - y_k\sum_{j=1}^{m}\frac{W_j}{N}\right)\right|\\ &= \gamma\left|y_k\sum_{j=1}^{m}\frac{W_j}{N} - x_k\sum_{j=1}^{m}\frac{W_j}{N}\right|\\ &= \gamma\sum_{j=1}^{m}\frac{W_j}{N}|y_k - x_k|\end{aligned} \tag{12}$$

Because $L = \max\left\{\gamma \sum_{j=1}^{m} \frac{W_j}{N}\right\}$, $|F_k(t,x) - F_k(t,y)| = \gamma \sum_{j=1}^{m} \frac{W_j}{N} |y_k - x_k| \leqslant L|y_k - x_k|$. Therefore, for any$k \in \{0, 1, 2\ldots, m\}$, $t \in [0, +\infty)$, the replicator dynamics Eq. (11) has a unique solution.□

Let $\dot{x}_k(t) = 0$ and we can get the unique solution X^* in formula (11). The solution process is as follows:

$$\begin{aligned}\dot{x}_k(t) = \gamma x_k(t)\left[\pi_k(t) - \bar{\pi}(x)\right] &= 0 = \gamma\frac{W_k}{N} - \gamma x_k\sum_{j=1}^{m}\frac{W_j}{N} = 0\\ &= \gamma/N\left[W_k - x_k\sum_{j=1}^{m} W_j\right] = 0\\ x_k &= \frac{W_k}{\sum_{j=1}^{m} W_j}\end{aligned} \tag{13}$$

According to formula (13), the unique solution of $\dot{x}_k(t)$ is $X^* = [x_1^*, \ldots, x_k^*, \ldots x_m^*]$, where $x_k^* = W_k / \sum_{j=1}^{m} W_j$.

Theorem 2. *For the multi-user computation offloading evolutionary game, starting from any non-zero state, the replicator dynamics can remain stable at the unique solution X^* proposed in theorem 1.*

Proof. According to the method of determining the stability of equilibrium point of differential equation, if $\dot{x}(t) = f(x)$ and $f'(x_0) < 0$, then x_0 is the equilibrium point of $\dot{x}(t)$, and if$f'(x_0) > 0$, then x_0 is not the equilibrium point of $\dot{x}(t)$. We derive the replicator dynamics Eq. (11) and we get $\ddot{x}(t) = -\gamma \sum_{j=1}^{m} \frac{W_j}{N}$, $\ddot{x}_k(t)$ is constant less than zero, so the replicator dynamic remains stable at X^*. □

Theorem 3. *The multi-user computation offloading evolutionary game has a unique evolutionary stability strategy (ESS).*

Proof. According to theorem 1, formula (11) has a globally unique solution X^*, and$x_k^* = W_k / \sum_{j=1}^{m} W_j$ is obtained when $\dot{x}_k(t) = 0$. Theorem 2 proves that Eq. (11) remains stable at X^*. Therefore, X^* can guarantee that the game reaches a balance point and remains stable, that is, $X^* = [x_1^*, \ldots, x_k^*, \ldots x_m^*]$ is the only ESS for the game proposed in this paper.□

5. Design of edge computation offloading method in distributed environment

In the previous section, we prove that there is a unique ESS in the game, but in the real scenario, due to the dynamic nature of the IoT device and the time-varying channel, it is difficult for devices to understand the strategies of other devices and their costs. It is difficult to reach the ESS in the game. In [62], it is pointed out that the learning model is consistent with the process of the replicator dynamics in continuous time. So we use reinforcement learning to implement ESS in a distributed environment. We propose a distributed evolutionary game algorithm based on reinforcement learning (EGT-QL), which does not require environment model and payment information of other devices. IoT device adjusts its offloading strategy according to the historical information that interacts with the environment, and gradually converges to the ESS.

5.1. Distributed evolutionary game algorithm based on reinforcement learning (EGT-QL)

We propose a distributed computation offloading method based on Q learning, where each device updates its own strategy according to action-reward independently, without interaction with other device. $S \times K$ matrix $R(t)$ represents the IoT device's immediate reward matrix. The elements in the matrix are calculated by formula (8). The strategy reward value with low consumption is larger, while the strategy reward value with high consumption is smaller. $S \times K$ matrix $Q(t)$ represents the Q value obtained by the device according to the reward of environment feedback after taking strategy K. λ and ζrepresent learning efficiency and discount factor. respectively. $S \times K$ matrix $Q_{\max}$ represents the state of convergence of the algorithm. $count(t)$ represents the actual number of iterations at time t. $count_{desired}(t)$ represents the expected number of iterations. Each IoT device stores a Q table and selects an offloading strategy based on the state-action pairs in the Q table. The proposed algorithm combines exploration and utilization. By introducing the greedy algorithm, the IoT device can obtain the optimal strategy in the current state, and in order to avoid the local maximization phenomenon, we introduce the exploration probability P. When $rand() \leqslant P$, the IoT device does not follow the optimal strategy chosen by the greedy algorithm but selects a strategy randomly. With continuous interaction with the environment, each device will eventually choose the optimal strategy. The pseudo code is as follows:

Algorithm (*1 Distributed evolutionary game algorithm based on Q-learning*).

```
1:Initialization:
    t = 0;
    For all $\forall i$, define the reward matrix as $R(0)$, the Q-value
    matrix$Q_{S,K}(0) = Q_{\max}(0) = 0$, and the number of counts as
    $count(0) = 0$, and set the value of $P$, $\lambda$, $\zeta$;
2:While (t>0)
3:      Choose a random strategy;
4:      if $(rand() \leqslant P)$
5:      Sensor i selects a action as $k(t)$ randomly;
6:      else
7:      Device i selects $\max(Q_{S,A}(t))$ as $k(t)$;
8:      end if
9:      Update the Q-value $Q_{S,K}(t)$, i.e.,
10:       $Q_{S,K}(t) = (1 - \lambda)Q_{S,K}(t) + \lambda(\pi_{k(t)}(t, k(t), X_{-k(t)})$
    $+\zeta \max Q_{i,k}(t))$
11:       if
    $(\sum_S \sum_K |Q_{\max} - Q_{S,K}(t)| < 0.0001$ and $\sum_S \sum_K Q_{S,K}(t) > 0$ )
12:         if $(count > count_{desired})$
13:           $Q_{\max} = Q_{S,K}(t)$;
14:             break;
15:         else
16:           $Q_{\max} = Q_{S,K}(t)$;
17:           $count = count + 1$;
18:         end if
19:       else
20:         $Q_{\max} = Q_{S,K}(t)$;
21:       $count = 0$;
22:       end if
23:end while
```

5.2. Convergence of evolutionary game theory based on Q learning (EGT-QL)

We firstly analyze the convergence of the algorithm under the evolutionary game model. The convergence characteristics are evaluated from two aspects: the convergence direction and convergence rate. Assuming that IoT device selecting strategy k at time t has the maximum expected utility, then the population distribution corresponding to this strategy $x_k(t) > 0$. k' represents the strategy selected by other devices, then $x_{k'}(t) > 0.d(x_k(t)/x_{k'}(t))/dt$ represents the rate of change of the ratio of the population distribution with k strategy and that with k' strategy over time t. The change trend of the replicator dynamics Eq. (11) is the same as $d(x_k(t)/x_{k'}(t))/dt$, and $d(x_k(t)/x_{k'}(t))/dt$ can be expressed as follows:

$$\begin{aligned}\frac{d\left(\frac{x_k(t)}{x_{k'}(t)}\right)}{dt} &= \frac{\dot{x_k}(t)\, x_{k'}(t) - x_k(t)\, \dot{x_{k'}}(t)}{\left(x_{k'}(t)\right)^2} \\ &= \frac{\gamma x_k(t)\left[\pi_k(t) - \bar{\pi}(x)\right] x_{k'}(t) - x_k(t)\gamma x_{k'}(t)\left[\pi_{k'}(t) - \bar{\pi}(x)\right]}{\left(x_{k'}(t)\right)^2} \\ &= \frac{\gamma x_k(t)\left[\pi_k(t) - \pi_{k'}(t)\right]}{x_{k'}(t)}\end{aligned} \tag{14}$$

It can be seen from formula (14) that $\pi_k(t) - \pi_{k'}(t) \geqslant 0$, so the replicator dynamics Eq. (11) develops in the direction of increasing the ratio of $x_k(t)$ to $x_{k'}(t)$. And from formula (14), we can get:

$$\begin{aligned}&\frac{d\left(\frac{x_k(t)}{x_{k'}(t)}\right)}{dt} \geqslant 0 \\ &\Rightarrow \dot{x_k}(t)x_{k'}(t) - x_k(t)\dot{x_{k'}}(t) \geqslant 0 \\ &\Rightarrow \frac{\dot{x_k}(t)}{x_k(t)} \geqslant \frac{\dot{x_{k'}}(t)}{x_{k'}(t)}\end{aligned} \tag{15}$$

According to Eqs. (14) and (15), the replicator dynamics Eq. (11) is a monotone function. Therefore, the higher the expected utility of strategy s_k is, the higher the proportion of s_k selected in the population will be. According to the literature [65], we can get that in continuous time, the learning model will generate multiple iterations at each time interval, and the adjustments made by the player between the two iterations of learning are very small. Therefore, according to the law of large numbers, this process becomes certain, and the learning model will eventually converge to the deterministic, continuous-time replicator dynamics process. Therefore, the convergence of the EGT-QL algorithm proposed based on the actual scenario is the same as the convergence under the replicator dynamics.

5.3. Algorithm complexity analysis

In this section, we will analyze the complexity of the algorithm from two aspects: space complexity and time complexity. We first discuss the space complexity, Q learning uses Q table $Q(S,K)$ to store data, where S represents the state space and K represents the action space. Therefore, the space complexity of the EGT-QL algorithm is $O(|S||K|)$. In addition, the computational complexity of EGT-QL algorithm is relatively low. Given a specific state, it's corresponding Q value can be obtained.

In terms of time complexity, currently, the use of Q learning in MDP model is relatively undeveloped in terms of time complexity [66]. Time complexity based on reinforcement learning requires a lot of mathematical analysis, which is not the focus of this paper. We evaluate the quality of offloading decisions by analyzing the convergence characteristics of EGT-QL algorithm after a certain number of iterations. As can be seen from Figs. 3 and 4, the energy consumption of the system decreases with the increase of the number of iterations, the utility increases gradually, and tends to be stable. In other words, the offloading strategy will be better and more stable with the increase of the number of iterations.

6. Experimental test and result analysis

In this section, we use Matlab to verify the convergence of EGT-QL and its performance in a dynamic environment. We establish a real scene network and deploy a set of IoT devices randomly within the coverage of the AP. And the time-varying channel follows Rayleigh fading. The main parameters required for the simulation are given in Table 1. The performances of the algorithm are evaluated mainly from three aspects: the convergence of the algorithm and the energy consumption and delay of the algorithm. Mainly compared with the following four algorithms:

Local computing: At each decision stage, IoT device chooses to perform computing tasks on its own device.

Edge server computing: In each decision stage, the device randomly selects the channel with the same probability to offload the task to the edge server.

Random assignment: Each IoT device randomly selects a offloading strategy and executes locally or at the server with the same probability.

BR-Algorithm (best-response algorithm)[55]: BR is implemented in an iterative manner. In each iteration, the BR algorithm allows to accept updated information to select the best strategy based on the instantaneous calculation cost, while other users keep their strategy unchanged.

6.1. Convergence analysis

In order to verify the convergence of EGT-QL, we divide the whole system into two populations, namely, performing computing tasks locally and performing computing tasks on the edge server. Assuming that the initial state of the two populations is [0.3, 0.7], we can see the evolution process and dynamic distribution of the population from Fig. 2. When the number of iterations reaches 25, the population distribution reaches convergence. When a device chooses a lower utility strategy, it will be motivated to choose another strategy to generate higher utility in the future evolution process. Therefore, the distribution of populations is a dynamic process, and the distribution of the two populations changes from [0.3, 0.7] in the initial state to ESS $x^* = [0.6, 0.4]$.

Figs. 3 and 4 show the relationship between system-wide computation cost, utility and number of iterations in a dynamic environment. At each iteration, IoT device selects the optimal strategy through policy update information, and joins the exploration probability to avoid local maximization. It can be seen from Fig. 3, system-wide computation cost gradually decreases and tends to be stable with the increase of the number of iterations. When the number of iterations reaches 1500, the system has the lowest computing consumption and tends to be stable.

Fig. 4 shows the relationship between the total utility of the system and the number of iterations. When the number of iterations increases, the overall utility of the system gradually increases and tends to be stable, and the utility of the system converges to a stable value when the number of iterations reaches 2000. It can be

Table 1
Simulation parameter setting.

Parameter	Value
Number of sensors N	[20,40]
Number of channels m	[1,10]
Bandwidth of channel B	5 MHz
Coverage of AP $d_{i,p}$	50 m
Transmit power p_i	0.1 w
Pass loss exponent α	4
Background noise σ_0	-100 dBm
Date size b_i	[5,40] Mb
Number of local computing CPU cycles d_i^{loc}	1000 Megacycles
Number of cloud computing CPU cycles d_i^{rem}	1200 Megacycles
Local computational capability F_i^{loc}	{0.5, 0.8, 1.0}GHz
Server computational capability F_i^{server}	12 GHz
Weight of computational time λ_i^t	{0, 0.5, 1.0}
Weight of computational energy λ_i^e	$1 - \lambda_i^t$
Computing energy efficiency $1/\eta_i$	{400, 500, 600}Megacycles/J
Utility constant β	10^3

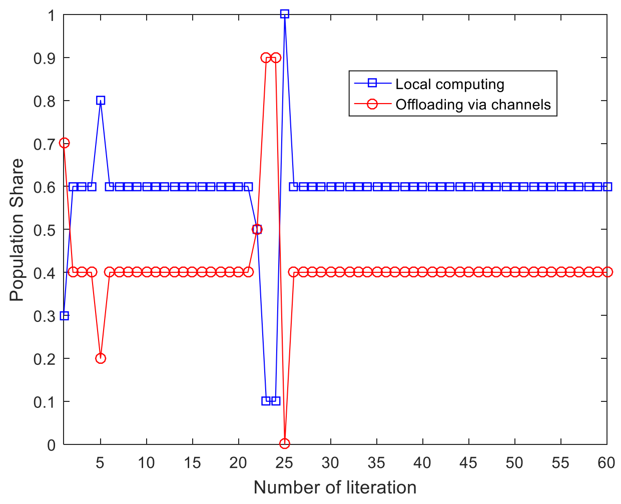


Fig. 2. Evolution of population share.

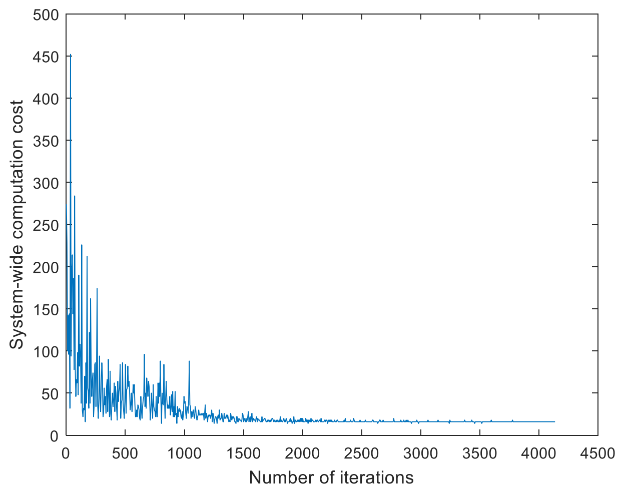


Fig. 3. System-wide computation consumption.

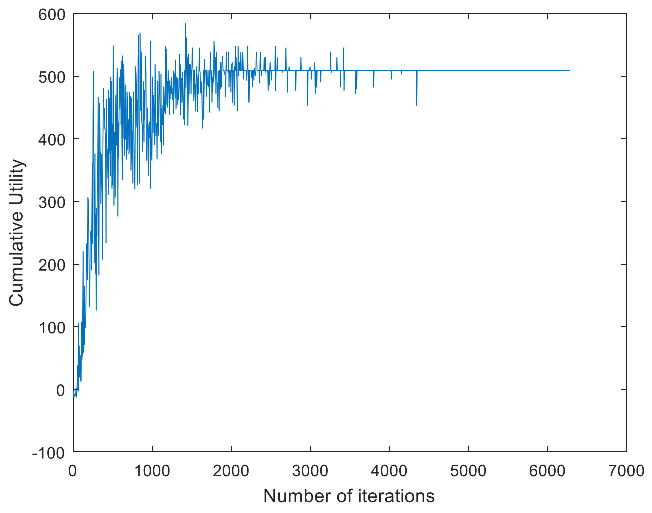


Fig. 4. System-wide utility.

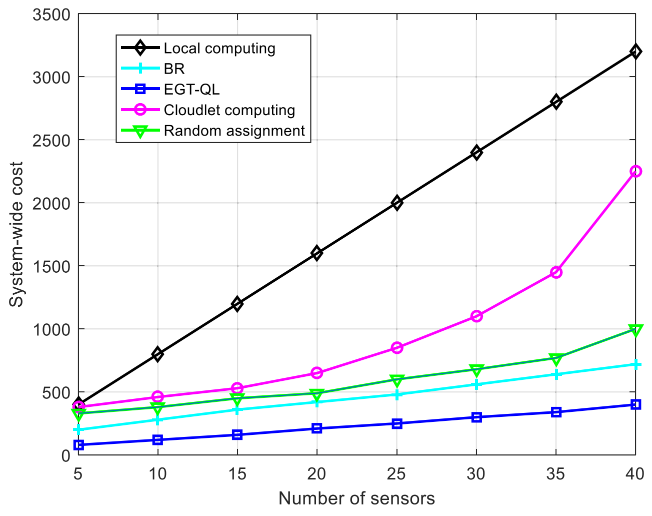


Fig. 5. System-wide computation cost of different IoT devices.

seen from Fig. 3 and Fig. 4 that the frequency of earlier changes in the system is relatively high, because the exploration P is relatively high. After several iterations, the frequency of system-wide computation cost and utility fluctuation becomes smaller due to the decrease of exploration P. At this time, IoT device performs exploitation more often. System-wide computation cost and utility converge to a stable value (ESS) as the number of iterations increases.

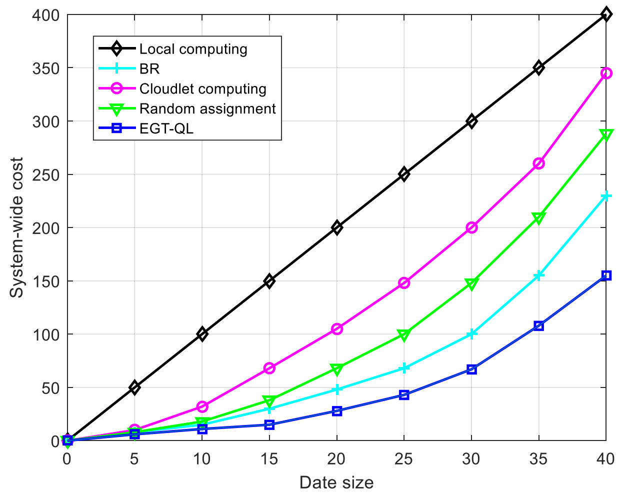


Fig. 6. System-wide computation cost for different packet sizes.

6.2. Algorithm comparison

In this section, we compare the performances of the EGT-QL, BR, local computing, edge server computing, and random assignment algorithms from system-wide computation cost and the average latency of the IoT device. Figs. 5 and 6 show the relationships between system-wide computation cost and the number of IoT devices and data size. As can be seen from Fig. 5, the consumption of the five algorithms increases as the number of devices increases. As the number of devices increases, the consumption of performing computing tasks at edge server increases at a faster rate. This is because the increase of the number of devices increases the congestion of wireless channels, which reduces the offloading efficiency of devices and increases system-wide computation cost. The total consumption of local computing increases linearly as the number of devices increases. The BR algorithm always chooses the optimal strategy based on the instantaneous cost, and does not change the offloading strategy of the device according to the dynamic environment. Therefore, as the number of sensors increases, the BR algorithm increases faster than our proposed algorithm, and the overall consumption is higher. And we can see from the figure that the algorithm EGT-QL proposed in this paper is significantly lower than other four algorithms in terms of system-wide computation cost. In Fig. 6, system-wide computation cost increases as the packet size increases. Moreover, our proposed algorithm consumes the minimum in the range of 0–40 Mb.

Fig. 7 shows the relationship between the number of channels and system-wide computation cost. It can be seen from the figure that when the number of channels is less than 3, the consumption of local calculation is less than that of random assignment and edge server execution. Because when the number of channels of the system is small, the computing resources are offloaded to the edge server at the same time, which will cause serious channel resource competition, which makes the transmission efficiency low and system-wide computation cost increases. Local computing

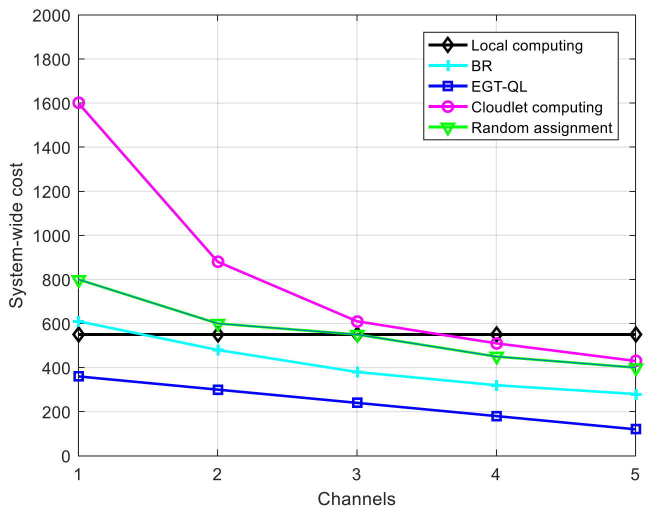


Fig. 7. System-wide computation cost of different channel counts.

does not need to transmit data over the channel, so the consumption of selecting local computing is independent of the number of channels. With the increase of the number of channels, the consumption of BR gradually decreases. However, BR does not change the offloading strategy of the device according to the dynamic environment. The total consumption of BR is higher than EGT-QL. The EGT-QL algorithm proposed in this paper has the lowest system-wide computation cost, because EGT-QL can select the optimal strategy according to the dynamic environment, reducing channel resource competition and improving transmission efficiency.

From Figs. 8 and 9, we can see the relationship between the average delay and the number of IoT devices and channels. As can be seen from Fig. 8, the average latency of four algorithms all increase with the increase of the number of devices. This is because the device's growth increases the burden of wireless channels, which reduces the offloading efficiency of devices and increases the transmission delay. Local computing does not require data to be transferred, so an increase in the number of devices does not affect the average latency of local computing. Our proposed algorithm will adapt the offloading strategy according to the changes of the environment, so the average delay of EGT-QL is smaller than that of the other three algorithms. It can be seen from Fig. 9 that when the number of channels increases, the average delay of EGT-QL, edge server computing, BR and random allocation algorithm gradually decreases. More channels can reduce the pressure of wireless communication and increase the transmission efficiency, thus reducing the average delay of equipment.

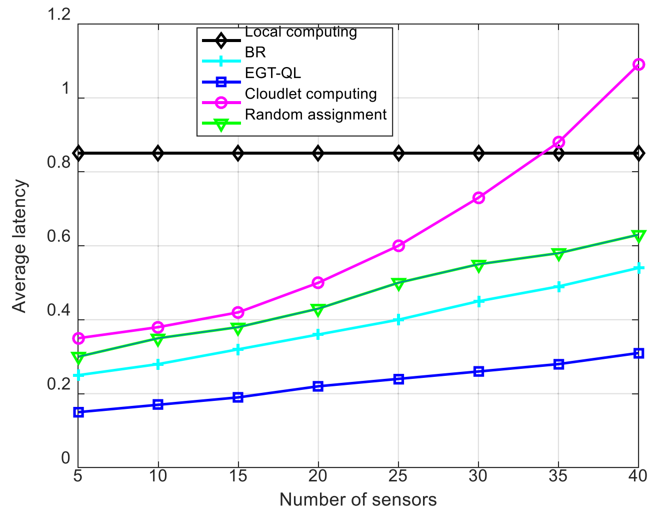


Fig. 8. Average delay for different number of IoT devices.

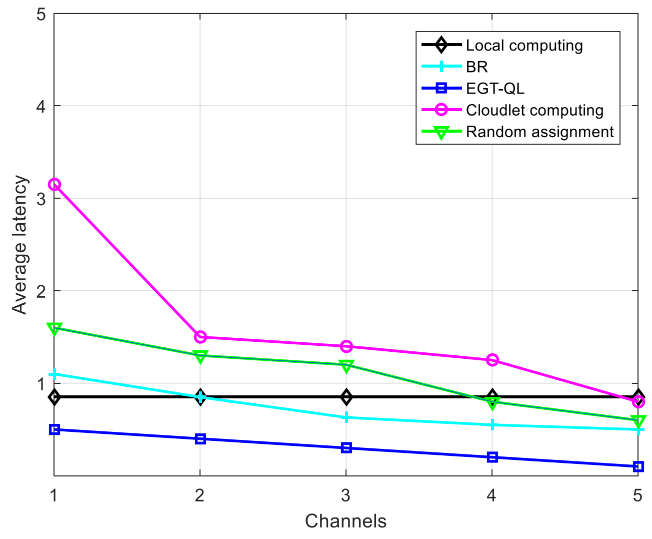


Fig. 9. Average delay of the number of different channels.

7. Conclusion

To solve the multiple IoT devices computation offloading problem in mobile edge computing, we propose a multi-user computation offloading method based on evolutionary game theory. First of all, we consider the multi-user computation offloading problem as an evolutionary game model. Then, we analyze the strategy selection process according to the replicator dynamics, and prove that the proposed evolutionary game model can achieve the unique ESS. At the same time, considering the practical application scenarios, we design an evolutionary game algorithm based on reinforcement learning, and analyze the convergence of EGT-QL algorithm. The experimental results show that the proposed algorithm can achieve stable convergence in terms of population distribution, system-wide computation consumption and utility, and average delay of IoT devices show better performance under dynamic environment.

In future work, we will consider a dynamic situation where the IoT device may leave the current edge server during computation offloading, in which case an effective movement model needs to be set up for the device, which will bring more technical challenges.

Declaration of Competing Interest

The authors declared that there is no conflict of interest.

References

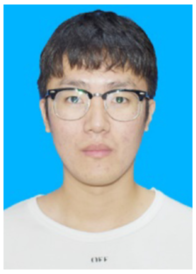
[1] Prachin B, Parul S. Communication technologies and security challenges for internet of things: a comprehensive review. Int J Electron Commun 2019;99(2):81–99.
[2] Zhang DG, Song XD. Extended AODV routing method based on distributed minimum transmission (DMT) for WSN. Int J Electron Commun 2015;69(1):371–81.
[3] Zhang T. Novel self-adaptive routing service algorithm for application of VANET. Appl Intellig 2019;49(5):1866–79. https://doi.org/10.1007/s10489-018-1368-y.
[4] Zheng K, Zhang T. A novel multicast routing method with minimum transmission for WSN of cloud computing service. Soft Comput 2015;19(7):1817–27.
[5] Zhang T. A kind of effective data aggregating method based on compressive sensing for wireless sensor network. EURASIP J Wirel Commun Netw 2018;2018(159):1–15.
[6] Zhang T. Novel optimized link state routing protocol based on quantum genetic strategy for mobile learning. J Netw Comput Appl 2018;2018(122):37–49.
[7] Zhang DG. New Multi-hop Clustering Algorithm for Vehicular Ad Hoc Networks. IEEE Trans Intell Transp Syst 2019;20(4):1517–30.
[8] Liu S. Novel unequal clustering routing protocol considering energy balancing based on network partition & distance for mobile education. J Netw Comput Appl 2017;88(15):1–9.
[9] Liu S. Novel Dynamic Source Routing Protocol (DSR) Based on Genetic Algorithm-Bacterial Foraging Optimization (GA-BFO). Int J Commun Syst 2018;31(18):1–20.
[10] Tang YM. novel reliable routing method for engineering of internet of vehicles based on graph theory. Eng Comput 2019;36(1):226–47.
[11] Gong CL, Jiang KW. A kind of new method of intelligent trust engineering metrics (ITEM) for application of mobile Ad Hoc network. Eng Comput 2019;11:1–13.
[12] Liu XH, Cui YY. A kind of novel RSAR protocol for mobile vehicular Ad hoc network. CCF Trans Netw 2019;2(2):111–25.
[13] Liu S. Adaptive repair algorithm for TORA routing protocol based on flood control strategy. Comput Commun 2020;151(1):437–48.
[14] Peng XH, Ren J, She L. BOAT: A block-streaming app execution scheme for lightweight IoT devices. IEEE Intern Things J 2018;5(3):1816–29.

[15] Wang X, Song XD. New medical image fusion approach with coding based on SCD in wireless sensor network. J Electr Eng Technol 2015;10(6):2384–92.
[16] Zhang XD. Design and implementation of embedded un-interruptible power supply system (EUPSS) for web-based mobile application. Enterprise Informat Syst 2012;6(4):473–89.
[17] Zhang DG. A new approach and system for attentive mobile learning based on seamless migration. Appl Intell 2012;36(1):75–89.
[18] Zheng K, Zhao DX. Novel Quick Start (QS) Method for Optimization of TCP. Wireless Netw 2016;22(1):211–22.
[19] Zhu YY. A new constructing approach for a weighted topology of wireless sensor networks based on local-world theory for the Internet of Things (IOT). Comput Math Appl 2012;64(5):1044–55.
[20] Zhou S. A low duty cycle efficient MAC protocol based on self-adaption and predictive strategy. Mobile Netw Appl 2018;23(4):828–39.
[21] Niu HL. Novel PEECR-based clustering routing approach. Soft Comput 2017;21(24):7313–23.
[22] Wang X, Song XD. New clustering routing method based on PECE for WSN. EURASIP J Wireless Commun Netw 2015;2015(162):1–13.
[23] Liu S. Dynamic analysis for the average shortest path length of mobile Ad Hoc networks under random failure scenarios. IEEE Access 2019;7:21343–58.
[24] Gao JX. Novel approach of distributed & adaptive trust metrics for MANET. Wireless Netw 2019;25(6):3587–603.
[25] Zhang T. A kind of novel method of power allocation with limited cross-tier interference for CRN. IEEE Access 2019;7(1):82571–83.
[26] Liu XH. A new algorithm of the best path selection based on machine learning. IEEE Access 2019;7(1):126913–28.
[27] Zhao PZ, Cui YY. A new method of mobile Ad Hoc network routing based on greed forwarding improvement strategy. IEEE Access 2019;7(1):158514–24.
[28] Yang JN, Mao GQ. Optimal base station antenna downtilt in downlink cellular networks. IEEE Trans Wireless Commun 2019;18(3):1779–91.
[29] Duan PB. A unified spatio-temporal model for short-term traffic flow prediction. IEEE Trans Intell Transp Syst 2019;20(9):3212–23.
[30] Chen JQ, Mao GQ. Capacity of cooperative vehicular networks with infrastructure support: multi-user case. IEEE Trans Veh Technol 2018;67(2):1546–60.
[31] Chen JQ. A topological approach to secure message dissemination in vehicular networks. IEEE Trans Intell Transp Syst 2020;21(1):135–48.
[32] Pan JL, Mcelhannon J. Future edge cloud and edge computing for internet of things applications. IEEE Internet Things J 2017;1(1):99.
[33] Zeng D, Gu L, Guo S. Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. IEEE Trans Comput 2016;65(12):3702–12.
[34] Hu YC, Patel M, Sabella D. Mobile edge computing: a key technology towards 5G. ETSI White Paper 2015;11(11):1–16.
[35] Pavel M, Zdenek B. Mobile edge computing: a survey on architecture and computation offloading. IEEE Commun Surveys Tutorials 2017;1(1):99.
[36] Beck MT, Maier M. Mobile edge computing: Challenges for future virtual network embedding algorithms. In: Proc. 8th Int. Conf. Adv. Eng. Comput. Appl. Sci. (ADVCOMP IARIA). p. 65–70.
[37] Chen XF, Zhang HG, Wu C. Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning. IEEE Internet Things J 2019;6(3):4005–18.
[38] Shi W, Cao J, Zhang Q. Edge computing: vision and challenges. IEEE Internet Things 2016;3(4):637–46.
[39] Mao Y, You C, Zhang J. A survey on mobile edge computing: the communication perspective. IEEE Commun Surveys Tuts 2017;19(4):2322–58.
[40] Mach P, Becvar Z. Mobile edge computing: a survey on architecture and computation offloading. IEEE Commun Surveys Tuts 2017;19(3):1628–56.
[41] You C, Huang K, Chae H. Energy efficient mobile cloud computing powered by wireless energy transfer. IEEE J Sel Areas Commun 2016;34(5):1757–71.
[42] Zhang W, Wen Y, Wu DO. Collaborative task execution in mobile cloud computing under a stochastic wireless channel. IEEE Trans Wireless Commun 2015;14(1):81–93.
[43] Wang C, Liang C, Yu FR. Computation offloading and resource allocation in wireless cellular networks with mobile edge computing. IEEE Trans Wireless Commun 2017;16(8):4924–38.
[44] Hu X, Wong KK, Yang K. Wireless powered cooperation-assisted mobile edge computing. IEEE Trans Wireless Commun 2018;17(4):2375–88.
[45] Wang F, Xu J, Wang X. Joint offloading and computing optimization in wireless powered mobile-edge computing systems. IEEE Trans Wireless Commun 2018;17(3):1784–97.
[46] Zhang DG, Chen C, Cui YY. New method of energy efficient subcarrier allocation based on evolutionary game theory. Mobile Netw Appl 2018. https://doi.org/10.1007/s11036-018-1123-y.
[47] Yu S, Wang X, Langar R. Computation offloading for mobile edge computing: a deep learning approach. 2017 IEEE 28th annual international symposium on personal, indoor, and mobile radio communications (PIMRC). IEEE; 2017.
[48] Zhang Y, Niyato D, Wang P. Offloading in mobile cloudlet systems with intermittent connectivity. IEEE Trans Mobile Comput 2015;Preprints(12):2516–29.
[49] Kwak J, Kim Y, Lee J. DREAM: Dynamic resource and task allocation for energy minimization in mobile cloud systems. IEEE J Sel Areas Commun 2015;33(12):2510–23.
[50] Huang D, Wang P, Niyato D. A dynamic offloading algorithm for mobile computing. IEEETransWireless Commun 2012;11(6):1991–5.
[51] Wang J, Peng J, Wei J. Adaptive application offloading decision and transmission scheduling for mobile cloud computing. In: Communications (ICC), 2016 IEEE international conference on. IEEE. p. 1–7.
[52] Yang L, Cao J, Tang S. A framework for partitioning and execution of data stream applications in mobile cloud computing. Proc IEEE Int Conf Cloud Computing 2012:794–802.
[53] Zhao Y, Zhou S, Zhao T. Energy-efficient task offloading for multiuser mobile cloud computing. In: Proc. IEEE/CICInt. Conf. On Commun. in China(ICCC). p. 1–5.
[54] Sardellitti S, Scutari G, Barbarossa S. Joint optimization of radio and computational resources for multi cell mobile-edge computing. IEEE Trans Signal Informat Process over Netw 2015;1(2):89–103.
[55] Chen X, Lei J. Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans Netw 2015;24(5):2795–808.
[56] Shermila R, Setareh M. Mobile Edge Computation Offloading Using Game Theory and Reinforcement Learning. arXiv preprint arXiv:1711.09012, Nov; 2017.
[57] Cao HJ, Cai J. Distributed multiuser computation offloading for cloudlet-based mobile cloud computing: a game-theoretic machine learning approach. IEEE Trans Veh Technol 2018;67(1):752–64.
[58] Zheng JC, Cai YM. Dynamic computation offloading for mobile cloud computing: a stochastic game-theoretic approach. IEEE Trans Mob Comput 2018;18(4):771–86.
[59] Quang DT, Duy LQ, Quek TQS. Distributed Learning for Computation Offloading in Mobile Edge Computing. IEEE Trans Commun 2018;66(12):6353–67.
[60] Cuervo E. MAUI: Making smart phones last longer with code offload. In: Proc. 8th Int. Conf. Mobile Syst., Appl., Services. p. 49–62.
[61] Yang L. A framework for partitioning and execution of data stream applications in mobile cloud computing. Perform Eval Rev 2013;40(4):23–32.
[62] Wen Y, Zhang W, Luo H. Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones. Proc IEEE Infocom 2012:2716–20.
[63] Zhang DG, Li G, Zheng K. An energy-balanced routing method based on forward-aware factor for Wireless Sensor Network. IEEE Trans Ind Inf 2014;10(1):766–73.
[64] Zhu K, Hossain E, Niyato D. Pricing, spectrum sharing, and SP selection in two-tier small cell networks: a hierarchical dynamic game approach. IEEE Trans Mob Comput 2014;13(8):1843–56.
[65] Börgers T, Sarin R. Learning through reinforcement and replicator dynamics. J Econ Theory 1997;77(1):1–14.
[66] Strehl AL. PAC model-free reinforcement learning. In: Proc. 23rd Int. Conf. Mach. Learn.. p. 881–8.

Yu-ya Cui (M'17) Born in 1992, Ph.D candidate, a Member (M) of IEEE in 2017. Now he is a researcher of Tianjin University of Technology, Tianjin, 300384, China. His research interest includes WSN, mobile computing, etc. His E-mail: 844511468@qq.com.

De-gan Zhang (M'01) Born in 1969. He received the Ph. D. degree from Northeastern University, Shenyang, China. He is currently a professor with Tianjin Key Lab of Intelligent Computing and Novel Software Technology, Key Lab of Computer Vision and System, Ministry of Education, Tianjin University of Technology, Tianjin 300384, China. His research interests include wireless sensor network and industrial applications. His E-mail: gandegande@126.com.

Ting Zhang (M'08) Born in 1973, Ph.D candidate, a Member (M) of IEEE in 2008. Now she is a researcher Tianjin University of Technology, Tianjin, 300384, China. Her research interest includes ITS, WSN, etc. Her E-mail: zhangtingts @163.com**.**



Ming-jie Piao (M'19) Born in 1995, Ph.D candidate, a Member (M) of IEEE in 2019. Now he is a researcher Tianjin University of Technology, Tianjin, 300384, China. His research interest includes ITS, WSN, etc. His E-mail: 849245164@qq.com.

Lu Chen (M'17) Born in 1991, Ph.D candidate, a Member (M) of IEEE in 2017. Now she is a researcher Tianjin University of Technology, Tianjin, 300384, China. Her research interest includes ITS, WSN, etc. Her E-mail: 1287725598@qq. com.

Hao-li Zhu (M'19) Born in 1995, Ph.D candidate, a Member (M) of IEEE in 2019. Now she is a researcher Tianjin University of Technology, Tianjin, 300384, China. Her research interest includes ITS, WSN, etc. Her E-mail:1819469356@qq.com.