ICC 2022 - IEEE International Conference on Communications | 978-1-5386-8347-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICC45855.2022.9838691

Dynamic Multi-user Computation Offloading for Mobile Edge Computing using Game Theory and Deep Reinforcement Learning

1st Peyvand Teymoori
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
pteymoor@uottawa.ca

2nd Azzedine Boukerche
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
aboukerche@uottawa.ca

***Abstract*—Mobile edge computing (MEC) has appeared as a promising solution to fill the gap between the growing computationally intensive applications and limited computation capability of mobile devices by providing powerful computing services at the edge of the wireless access network. To use the services provided by the MEC more effectively, making efficient and reasonable offloading decisions is crucial. In this paper, we study the computation offloading of tasks from multiple users to a single-cell edge server under a dynamic environment. We consider a practical case wherein a group of mobile users with random mobility patterns use a common set of time-varying stochastic transmission channels to perform computation offloading, and the number of active users in the system randomly changes. To reduce the mutual interference among users when accessing the wireless channels, we adopt game theory to formulate the users' computation offloading decision process as a stochastic game model. Next, we prove the existence of the Nash Equilibrium (NE) for the proposed game model by showing its equivalency to a weighted potential game which has at least one pure-strategy NE point. Then, we present distributed computation offloading algorithms by adopting a payoff-based multi-agent reinforcement learning (MARL) approach to reach the NE of the game. Finally, through simulation, we validate the effectiveness of the proposed algorithms by comparing them with the results obtained from other previously studied multi-agent learning algorithms as well as conventional Q-learning and deep Q-learning algorithms.**

***Index Terms*—Mobile edge computation, multi-user computation offloading, game theory, Multi-agent reinforcement learning.**

I. Introduction

As mobile devices are gaining enormous popularity, advancements in computing hardware and communication technologies have enabled them to support more complex applications such as face recognition, interactive gaming, natural language processing (NLP), and augmented reality (AR). These sophisticated applications usually require significant amounts of computation and energy resources, which, however, cannot be directly afforded by most mobile devices due to their limited computation capabilities and battery supply.

This work is partially supported by NSERC, Canada Research Chairs Program, and NSERC-CREATE TRANSIT program.

To address this challenge, Mobile Cloud Computing (MCC) is proposed as an effective approach to release the heavy workload on mobile devices and prolong their battery life. In MCC, a mobile device can offload a part of its computational tasks to be executed remotely on resource-rich cloud infrastructur [1]. Offloading computation tasks via wireless channels involves considerable communications between the mobile device and the remote could which may increase the time and energy overhead of the system. However, with the massive growth in intelligent and mobile devices coupled with technologies like Internet of Things (IoT), the focus has shifted towards achieving real-time responses along with support for context-awareness and mobility, which has encouraged the service providers to bring the computing servers to proximity of users. This concept is known as mobile edge computing (MEC).

The computation offloading problem from the perspective of a single mobile user has been extensively studied [2]. However, research efforts for the case of multi-user computation offloading is still limited. Since most wireless networks are multi-channel, a key problem in the case of multi-user offloading is how to achieve efficient wireless access coordination among multiple users to reduce the mutual interference caused by sharing a common set of channels [3]. Majority of studies in the area of multi-user computation offloading have dealt with this problem using centralized mechanisms which do not consider the interactions among multiple self-organizing users [4] [5]. In a multi-agent system, when agents have opposing goals, a clear optimal solution may no longer exist. In this case, an equilibrium between users' action selection strategies is usually searched for. Game theory is a widely adopted mathematical tool to model and analyze complicated decision-making processes among a group of rational and selfish decision-makers to find an equilibrium between users' strategies [6] [7]. In such an equilibrium, which is called the Nash Equilibrium (NE) point, no agent can improve its payoff when the other agents keep their actions fixed.

As different mobile devices are usually owned by different self-interested users, game theory can be utilized in the context of multi-user MEC to achieve an effective coordination among users to access the channels. [8] [9] proposed decentralized mechanisms based on game theoretic approaches to solve the multi-user computation offloading problems. These studies were mainly focused on the computation offloading problems under quasi-static environments where the set of mobile device users and the wireless channel conditions remain unchanged during a computation offloading. However, In a real scenario, due to the random environmental factors, the utility of each player is dynamically varying, and therefore, it may never reach the equilibrium solution of the static game model.

In this paper, we consider the problem of interference-aware multi-user computation offloading under dynamic environment, where the wireless communication channels are time-varying, and users who are following random mobility patterns can dynamically become active/idle or enter/leave the coverage area of AP. For this problem, we adopt the theory of general-sum Markov games, also known as stochastic games [10], which considers all the possible states of the dynamic process, to derive a multi-user computation offloading game model. We show that the proposed game model is a potential game which can achieve at least one pure-strategy NE.

Then, we utilize the concept of reinforcement learning (RL) to design computation offloading algorithms to reach the NE under dynamic environment. Various RL techniques have been developed which enables an agent to make a sequence of decisions to optimize its behaviour in a wide range of circumstances. However, when multiple learners simultaneously apply RL, even in a stationary environment, many complexities arise which make the traditional approaches often fail. When, in addition to multiple users, we assume a dynamic environment which requires multiple sequential decisions, the problem becomes even more complex since besides having to coordinates with other agents, now they also have to take into account the current state of the environment [11]. In this paper, we use a combination of RL and Game theory to develop a Multi-Agent Reinforcement Learning (MARL) algorithms, which we call Nash-Q-Learning. The proposed algorithm considers the general Markov game as a sequence of static games[1], known as stage game, at different time slots and uses the NE of each current stage game from the a current joint action payoff matrix, to find its way towards the NE point of the general Markov game. Moreover, in order to avoid Nash-Q-Learning algorithm suffering from dimensional disaster, we develop a deep reinforcement learning algorithm, referred to as Nash-DQN, by adding the neural networks to Nash-Q-Learning. We also use computer simulation to compare the effectiveness of the proposed MARL algorithms with conventional Q-learning and deep Q-learning algorithm as well as another previously studied multi-agent stochastic learning (MASL) algorithm from [10]. The simulation results show that the performance the proposed algorithms is superior to the other implemented multi-user computation offloading algorithms.

The rest of this paper is organized as follows. Section II introduces the system model. In Section III, we propose a general-sum Markov game to model the dynamic multi-user computation offloading problem. Then, in section IV, we prove the existence of the NE. Following that, in Section V, we introduce the proposed MARL algorithms to reach the NE of the Markov game. Numerical simulations are provided in section VI. Finally, we present our conclusions in Section VII.

[1]The system is taken to be stationary during the period of one time slot.

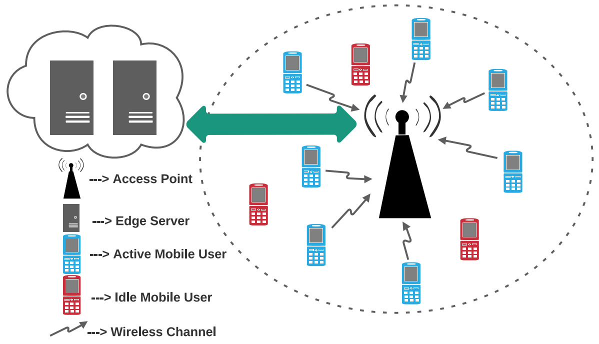

Fig. 1: MEC system model

II. System Model

We formulate the problem of multi-user computation offloading in a single-cell scenario. As indicated in Figure 1, There exists a wireless access point (AP) through which the mobile users can offload their computational tasks to a connected high performance MEC server. We consider a group of *active* mobile users $\mathcal{N} = \{1, 2, \ldots, N\}$ in the system. A mobile user is referred to as an *active user* if it has a released computation task to be executed. Otherwise, we call it an *idle user*. Note that, since the users randomly become active or idle and dynamically leave or enter the coverage area of the AP, the set $\mathcal{N}$ can change within different time slots. However, we assume that it remains fixed in the period of one time slot. Each active user n has a computation-intensive task K_n that needs to be executed either locally on the mobile device or remotely on a MEC server through computation offloading. The computation task can be expressed as $K_n = (B_n, D_n)$. Where, B_n denotes the size of computation input data size of the task K_n, and D_n shows the number of required CPU cycles to accomplish the computation task.

In this paper, we consider a general case that the wireless transmission channel from user n to AP is a stochastic channel which varies from time slot to time slot. To model the time-varying wireless channels, we assume that the channels between mobile users and the AP follow Rayleigh fading, which is a realistic and widely adopted channel model [12]. Specifically, at each time slot the instantaneous channel power gain from user n to the AP is given by $h_{t,n} = {d_{t,n}}^{-\alpha} \cdot \beta_{t,n}$, where $d_{t,n}$ is the distance from user n to the AP, α is the path loss exponent, and $\beta_{t,n}$ is the random Rayleigh fading factor. Notice that, the instantaneous random coefficient $\beta_{t,n}$ changes from time slot to time slot.

We assume a set of M available wireless transmission channels between the user and the base station which denoted as $\mathcal{M} = \{1, 2, \ldots, M\}$. Mobile user n's computation offloading strategies can be expressed as $a_n^t \in A_n = \{0\} \cup \mathcal{M}$, where A_n indicates the set of strategies for user n. Specifically, at each time slot t, $a_n^t > 0$ means that user n chooses to offload the computing task to the MEC server through the wireless channel $m \in \mathcal{M}$, while $a_n^t = 0$ indicates that user n decides to process the task locally. The number of uploaded bits at each time slot for each user depends on the instantaneous uplink data transmission rate of the selected channel. According to the Shannon principle, given the strategy profile $\mathbf{a}^t = [a_n]_{\forall n \in \mathcal{N}}$, the instantaneous uplink channel capacity for user $n \in \mathcal{N}$ under M orthogonal sub-channels, can be derived as follows:

$$R_n(\mathbf{a}^t, \mathcal{N}) = W_M log_2 \left(1 + \frac{p_n h_{t,n}}{\eta_0 + \sum_{i \in \mathcal{N} \setminus \{n\}: a_i = a_n} p_i h_{t,i}}\right) \tag{1}$$

W_M is the channel bandwidth, p_n is the transmit power of user n, and η_0 denotes the background noise power. As shown in (1), if too many active users share the same channel to simultaneously offload their computation to the edge server, they may incur severe interference to each other, resulting in low offloading rates, and therefore, more communication cost (i.e., longer transmission time and higher energy consumption), which compromises the benefits of using MEC. In this case, it would be more beneficial for the mobile user to compute the task solely locally without triggering any computation offloading. According to the offloading strategy profile $\mathbf{a}^t$, the delay and energy consumption for each user can be expressed as either local or remote computing. We next introduce them in detail.

A. Local Computing

When the mobile user decides to execute the task K_n locally by itself, the required amount of time and energy to process the tasks can be calculated by $T_n^L = \frac{D_n}{f_n^L}$ and $E_n^L = e_n^L \cdot D_n$, respectively. Where, f_n^L denotes the computation capability (i.e., CPU cycles per second) of mobile user n, and e_n^L is the energy per CPU cycle consumed by the mobile device.

Therefore, the total communication overhead to execute the task locally can be expressed by:

$$Q_n^L = \gamma_n^E E_n^L + \gamma_n^T T_n^L \tag{2}$$

where γ_n^E and $\gamma_n^T \in (0, 1)$ denote the weights of computational time and energy for mobile user n, respectively. These weights can be adjusted by the user itself according to its own priorities and the features or constraints of the computational task.

B. Remote Computing

If the active user n decides to offload its computational job k_n to the cloud, it would incur the cost for transmitting the input data to the edge server through the AP. For a given state of the environment at time slot t, $x_t = (\mathcal{N}, \mathbf{h}_t)$, where $\mathbf{h}_t = [h_{n,t}]_{\forall n \in \mathcal{N}}$, the transmission time and energy consumption of the mobile device for uploading the input data of size B_n are respectively given by $T_n^{up}(\mathbf{a}^t, x_t) = \frac{B_n}{R_n(\mathbf{a}^t, x_t)}$ and $E_n^{up}(\mathbf{a}^t, x_t) = \frac{p_n B_n}{R_n(\mathbf{a}^t, x_t)}$. Note that since the data size of the downloaded data is negligible compared to the size of the uploaded data, we can ignore the energy and time components to transmit the data from the edge server to the mobile device. Therefore, the total communication overhead to perform computation offloading can be expressed by:

$$\begin{aligned} Q_n^{Off}(\mathbf{a}^t, x_t) &= \gamma_n^E E_n^{up}(\mathbf{a}^t, x_t) + \gamma_n^T T_n^{up}(\mathbf{a}^t, x_t) \\ &= \frac{(\gamma_n^E p_n + \gamma_n^T) B_n}{R_n(\mathbf{a}^t, x_t)}, \quad n \in \mathcal{N}, a_n^t > 0 \end{aligned} \tag{3}$$

From (3) we see that, when user n decides to perform computation offloading, its cost is effected not only by the randomness caused by the non-stationary environment and its own offloading strategy, but also by the action-selection strategy of all the other active users. Due to such an interdependency among different mobile users, game theory is a suitable mathematical tool to model and analyze users' decision-making for computation offloading.

III. Problem Formulation and Stochastic Game Model

In this section, we formulate the problem of multi-user computation offloading in the MEC system as an optimization problem, in which each mobile user independently adjusts its computation offloading strategy to minimize its own computation offloading overhead. Given the task offloading strategy of all active users except user n, $\mathbf{a}_{-n}^t = [a_i^t]_{\forall i \in \mathcal{N}/n}$, and the current state of the environment, the computation overhead function for user n at time slot t is expressed by:

$$V_n(a_n^t, \mathbf{a}_{-n}^t, x_t) = \begin{cases} Q_n^L, & a_n^t = 0 \\ Q_n^{Off}(\mathbf{a}_{-n}^t, x_t), & a_n^t > 0 \end{cases} \tag{4}$$

To minimize the system-wide computation overhead, the following optimization problems should be solved simultaneously for each active user at each current time slot:

$$\min_{a_n^t} V_n(a_n^t, \mathbf{a}_{-n}^t, x_t), \tag{5}$$

The centralized optimization problem to minimize the system computation overhead is inherently an NP-hard problem. Therefore, we utilize game theory to solve the problem in a distributed manner. Note that for a given state of the environment at the current time slot t, $x_t = (\mathcal{N}, \mathbf{h}_t)$, the minimization problem can be modelled as a static game. However, in a dynamic case that environmental conditions vary from time slot to time slot, static model can not guarantee to converge to an optimal solution. Therefore, we propose a multi-user Markov game model $\Gamma = \{\mathcal{N}, X, \{A_n\}_{\forall n \in \mathcal{N}}, \{\bar{V}_n\}_{\forall n \in \mathcal{N}}\}$ to achieve an optimal decision-making process for mobile user's computation offloading. For each user n, we define

$\bar{V}_n(a_n, \mathbf{a}_{-n}) = \mathbb{E}_X[V_n(a_n, \mathbf{a}_{-n}, X)]$ as an expected payoff function over the set of all possible states of the environment, X. Based on the proposed game model, each mobile user independently adjusts its strategy to minimize its expected payoff function.

IV. Analysis of Nash Equilibrium

In game theory, Nash Equilibrium(NE) is the most important solution concept for analyzing the outcome of the strategic interaction of multiple decision-makers. NE in the game theory encourages the effective sharing of resources rather than fierce competition [13]. Therefore, the NE of the multi-user computation offloading game can effectively reduce the mutual interference and lead each user to reach a stable state of mutual satisfaction. In this section we prove the existence of NE in the stochastic game. To proceed, we first introduce the definition of NE for the game model.

Definition 1. *A computation offloading strategy $a^* = [a_n^*]_{\forall n \in \mathcal{N}}$ profile is an expected (pure-strategy) NE of the Markov game if and only if no mobile user can minimize its expected payoff function $\bar{V}_n$ by deviating unilaterally, i.e.,*

$$\bar{V}_n(a_n^*, \boldsymbol{a}_{-n}^*) \leq \bar{V}_n(a_n, \boldsymbol{a}_{-n}^*), \forall n \in \mathcal{N}, \ \forall a_n \in A_n \quad (6)$$

Definition 2. *A Markov game with an expected payoff function $\bar{V}_n$ is an exact potential game if there exists an expected potential function $\bar{\phi}$ such that for $\forall n \in \mathcal{N}$ and $\forall a_n, a_n' \in \boldsymbol{A}_{-n}$ the following conditions hold:*

$$\bar{V}_n(a_n', \boldsymbol{A}_{-n}) - \bar{V}_n(a_n, \boldsymbol{A}_{-n}) = \bar{\phi}_n(a_n', \boldsymbol{A}_{-n}) - \bar{\phi}_n(a_n, \boldsymbol{A}_{-n}) \quad (7)$$

Theorem 1. *The Markov game Γ is a weighted potential game with the expected potential function given by:*

$$\bar{\phi}(\boldsymbol{A}_{\mathcal{N}}) = \mathbb{E}_X[\phi(\boldsymbol{A}_{\mathcal{N}}, X)] = \frac{1}{2}\sum_{n=1}^{N}\sum_{n\neq i}\zeta_n\zeta_i p_n\bar{h}_n p_i\bar{h}_i l_{\{a_n=a_i\}} \quad (8)$$
$$l_{\{a_n>0\}} + \sum_{n=1}^{N}\zeta_i p_i\bar{h}_i l_{\{a_n=0\}}$$

where $\boldsymbol{A}_{\mathcal{N}}$ denotes the strategy profile of all the mobile users, ζ_n is the active probability of mobile user $n \in \mathcal{N}$, $\bar{h}_n$ is the expected channel gain from mobile user i to AP.

As proved in [14], every weighted potential game possesses the finite improvement property, and thus has at least one NE. Therefore, the investigated mobile users' decision making process is guaranteed to have at least one equilibrium solution.

V. Multi-Agent Reinforcement Learning Under Dynamic Environment

In this section, we use *MARL* framework to design distributed yet efficient algorithms to reach the NE of our proposed Markov game Γ under dynamic environment. Reinforcement Learning was originally developed for Markov Decision Processes (MDPs). It allows a single agent to learn a policy that maximizes a possibly delayed reward signal in a stochastic environment. However, when multiple agents apply reinforcement learning in a shared environment, this might be beyond the scope of the MDP model. In a multi-agent context, as opposed to a single-agent scenario, the immediate reward received by the agent n at each time slot depends not only on the system state x_t and the action a_n^t taken by that specific user, but also by the joint action of all other agents ($\mathbf{a}_{-n}^t$). In what follows, we propose two MARL algorithms, referred to as Nash-Q-Learning and Nash-DQN, which perform the update rule based on the NE of the stage game at each time slot to find its way toward the NE of the general-sum Markov game.

A. Multi-Agent Nash-Q-Learning

A number of approaches have been developed, aiming at extending the successful Q-learning algorithm to multi-agent systems [11]. Whereas it is possible to apply Q-learning in a straightforward fashion to each agent in a multi-agent system, doing so ignores a key issue, related to the multi-agent context, that the non-stationarity of the environment is not only generated by an arbitrary stochastic process, but rather by other rational agents who are adapting at each time slot.

To deal with this problem we propose a multi-agent algorithm based on Q-learning and NE (i.e., Nash-Q-Learning). In this algorithm, each user estimates the future payoff for all other users for any possible joint action, $\mathbf{a}$, where $\mathbf{a} = [a_n]_{\forall n \in \mathcal{N}}$, and forms a joint action payoff matrix at each stage game. So, as opposed the single agent Q-learning which uses the max operator to update the agent's own Q-table, Nash-Q-Learning uses a NE of the current stage game to perform the update rule. In the proposed algorithm, we also use ϵ-greedy method as the strategy of action exploration. With the ϵ-greedy method, the agent selects the optimal action according to the corresponding NE points of the stage game with probability 1-ϵ, and chooses a random action with probability $\epsilon \in [0, 1]$.

Algorithm 1 Multi-agent ϵ-greedy Nash-Q-Learning

```
1: Initialize: For all $x \in X$ and $a_n \in A_n$, $n \in \mathcal{N}$, Let $Q_n^0(x, a_1, a_2, \cdots, a_N) = 0$
2: while expected state $x_{terminal}$ is not reached do
3:     for Agents $n \in \mathcal{N}$ do
4:         With probability $\epsilon$ select a NE action, $a_n^t$ from the joint action payoff matrix, otherwise explore a random action $a_n^t$
5:         Observe $r_1^t, r_2^t, \cdots, r_N^t$ and $a_1^t, a_2^t, \cdots, a_N^t$ and $x_{t+1} = x'$
6:         Update agent n's joint-actions Q-table using (9)
7:         Let $x \leftarrow x'$
8:     end for
9:     Let t:=t+1
10: end while
```

According to section IV, we know that the proposed multi-user computation offloading game model can achieve a NE.

Algorithm 1 shows the exploration and learning process of the proposed Nash-Q-Learning algorithm. Note that in this algorithm, $Q_n^t(x, \mathbf{a})$ can be seen as a joint payoff matrix, in which each entry, $(Q_1^{t,n}, Q_2^{t,n}, \cdots, Q_N^{t,n})$, is a tuple consist of the the expected future payoff for all users which is estimated by user n at time slot t. At each training episode and for each user n, the joint action Q-table gets updated according to (9). Where, $NashQ_n^t(x', \mathbf{a})$ refers to the expected joint actions payoff tuple from the set of all NE points of the stage game, when being at state $x' = x_{t+1}$, that has the highest payoff for user n. α and β represent the learning rate and the discount factor the expected future rewards.

$$Q_n^{t+1}(x_t, \mathbf{a}) = (1-\alpha)Q_n^t(x, \mathbf{a}) + \alpha[r_n^t + \beta NashQ_n^t(x', \mathbf{a})] \tag{9}$$

B. Multi-Agent Nash Deep Q Network (MA-Nash-DQN)

In the multi-agent Nash-Q-Learning algorithm, each state-joint action tuple has corresponding vector of joint Q-values stored in the Q-table. In a real scenario, the computation offloading problem is very complex with a large number of tuples. If all joint Q-values are stored in tables, the memory will not meet the demand. In order to avoid Q-learning suffering from time and space complexity, instead of storing the Q-values in a table, a Q-function approximator which employ deep neural networks can be used. This is the Deep Q Network (DQN) basic idea.

Algorithm 2 Multi-agent ϵ-greedy Nash Deep Q-learning

```
1: Initialize: Replay memory $D$ to capacity $N$, joint Action-Value network $Q$ with random weights $\theta$, and joint Action-Value network $\hat{Q}$ with random weights $\hat{\theta} = \theta$.
2: while expected state $x_{terminal}$ is not reached do
3:     for Agents $n \in \mathcal{N}$ do
4:         With probability $\epsilon$ select an NE action, $a_n^t$, from the set of all NE points of the current stage game, otherwise explore a random action $a_n^t$.
5:         Execute action $a_n^t$ and Observe $r_1^t, r_2^t, \cdots, r_N^t$ and $a_n, \mathbf{a}_{-n}$ and $x_{t+1}$
6:         store the experience $e_n^t = (x_t, \mathbf{a}_t, \mathbf{r}_t, x_{t+1})$ in $D$
7:         Sample a random mini-batch of experiences $e_n^t = (x_j, \mathbf{a}_j, \mathbf{r}_j, x_{j+1})$ from $D$
8:         Set $y_j = \begin{cases} r_j, \text{if episode terminates at step } j+1, \\ r_j + Nash\hat{Q}(x_{j+1}, \mathbf{a}; \hat{\theta}) \quad \text{otherwise.} \end{cases}$
9:         Perform a gradient decsent step on $(y_j - Q((x_j, \mathbf{a}_j; \theta))^2$ with respect to network parameter $\theta$.
10:        Every C steps reset $\hat{Q} = Q$
11:    end for
12:    Let t:=t+1
13: end while
```

DQN takes the states as inputs and estimate the vector of joint Q-values for each corresponding joint action in the output layer. In DQN, the learning process uses two separate neural networks which are called main and target networks. These networks have the same architecture but different weights. Every N steps, the weights from the main network (θ) are copied to the target network ($\hat{\theta}$).

VI. Simulation Results

In this section, computer simulation is used to study the performance of the proposed MARL algorithms (e.g., Nash-Q-Learning, Nash-DQN) under dynamic environment. We use the same parameter setting as in [10]. We assume that the bandwidth of each channel is $W_M = 5MHz$, and users are randomly scattered within 50 meters from the AP. The random fading coefficient β is considered to be exponentially distributed with unit-mean. The computing power provided by the MEC server to users is $F^c = 10$ GHz. The number of mobile users and the number of available channels are 30 and 5, respectively. For the Gaussian white noise, we define $\eta_0 = -100$dBm. The local computational capacity and the transmission power for each user, are set to $f^l = 1$ GHz and $p = 100$ mw, respectively. The size of the computational task is 5 MB, and the number of CPU cycles required to execute the task is 1000 Megacycles. For simplicity, the weights of time and energy components are set to $\gamma_n^E = \gamma_n^T = 0.5$.

We validate the effectiveness of the proposed MARL algorithms by comparing them with traditional Q-learning and Deep Q-learning algorithms as well as two other previously studied algorithms referred to as *Multi-Agent Stochastic Learning (MASL)* and *Best Response (BR)* algorithms [10]. We also plot two other intuitive algorithms, referred to as *Local Execution* and *Random Strategy Selection (RSS)* algorithms.

To implement the traditional Q-learning and DQN algorithms in a multi-agent environment, we assume that each mobile user runs the update procedure independently and simultaneously, by considering other user's action as a part of the stochastic environment, to learn an individual optimal strategy. The Local Execution algorithm executes the entire job locally without doing any offloading. In RSS algorithm, each mobile user randomly selects a strategy at each time slot.

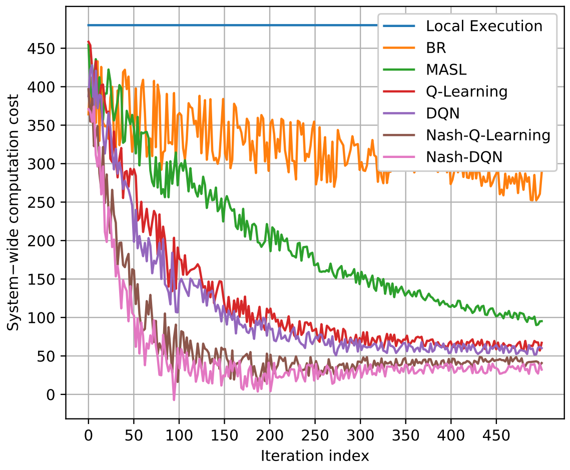

Fig. 2: system-wide cost vs. the number of iterations.

Figure (2) shows the system-wide computation cost versus the number of iterations for the above-mentioned algorithms.

As can be seen from this figure, for all learning algorithms, as the number of training episodes increases the total system overhead decreases. However, this reduction is much higher for the proposed NE based algorithms, and they converge greatly faster compared to other learning based algorithms. For Local Execution algorithm, since the delay and energy consumption only depend on the characteristics of the computation task and the computational capabilities of the mobile device, the system overhead keeps constant as the number of iteration increases.

Figure (3) shows the system-wide computation cost of algorithms for different number of users. As illustrated in this figure, as the number of users in the system increases, the total system overhead increases for all algorithms. However, the proposed NE based algorithms always consume a significantly less cost, compared to the other approaches.

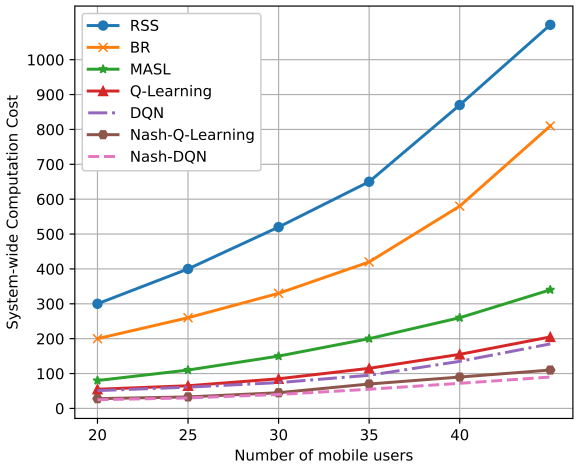

Fig. 3: System-wide cost vs. the number of active users.

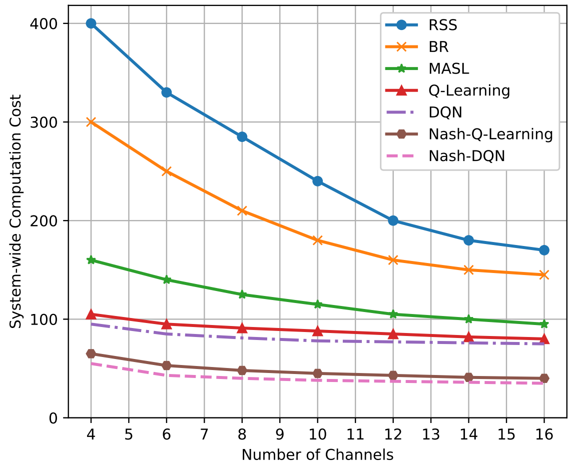

Fig. 4: System-wide cost vs. the number of available channels.

Figure (4) again shows that our proposed algorithms always outperforms all other algorithms. This is specifically true, when the number of available wireless channels is small.

VII. Conclusion

In this paper, we considered the problem of multi-user computation offloading for MEC under dynamic environment. We formulated the optimization problem as a Markov game model, and proved that mobile users' dynamic offloading decision process leads to a NE. To reach the NE, we proposed a fully distributed MARL algorithms based on NE of stage games, and investigated their convergence under dynamic environment. Simulation results showed that our proposed algorithm performs significantly better when compared to the other previously studied learning algorithms. The main problem left open for future work is the case when each user's task execution response time is subjected to a strict deadline.

References

[1] S. Guan and A. Boukerche, "Design and implementation of offloading and resource management techniques in a mobile cloud environment," in *Proceedings of the 17th ACM International Symposium on Mobility Management and Wireless Access*, 2019, pp. 97–102.

[2] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *Wireless Communications, IEEE Transactions on*, vol. 11, no. 6, pp. 1991–1995, 2012.

[3] A. Boukerche, K. El-Khatib, and T. Huang, "A performance comparison of dynamic channel and ressource allocation protocols for mobile cellular networks," in *Proceedings of the second international workshop on Mobility management & wireless access protocols*, 2004, pp. 27–34.

[4] Y. Zhao, S. Zhou, T. Zhao, and Z. Niu, "Energy-efficient task offloading for multiuser mobile cloud computing," in *2015 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, 2015, pp. 1–5.

[5] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, 2015.

[6] J. Zheng, Y. Cai, N. Lu, Y. Xu, and X. Shen, "Stochastic game-theoretic spectrum access in distributed and dynamic environment," *IEEE transactions on vehicular technology*, vol. 64, no. 10, pp. 4807–4820, 2014.

[7] D. Zheng, F. R. Yu, and A. Boukerche, "Security and quality of service (qos) co-design using game theory in cooperative wireless ad hoc networks," in *Proceedings of the second ACM international symposium on Design and analysis of intelligent vehicular networks and applications*, 2012, pp. 139–146.

[8] X. Ma, C. Lin, X. Xiang, and C. Chen, "Game-theoretic analysis of computation offloading for cloudlet-based mobile cloud computing," in *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2015, pp. 271–278.

[9] T. Lin, T. Alpcan, and K. Hinton, "A game-theoretic analysis of energy efficiency and performance for cloud computing in communication networks," *IEEE Systems Journal*, vol. 11, no. 2, pp. 649–660, 2015.

[10] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Transactions on Mobile Computing*, vol. 18, no. 4, pp. 771–786, 2018.

[11] A. Nowé, P. Vrancx, and Y.-M. De Hauwere, "Game theory and multi-agent reinforcement learning," in *Reinforcement Learning*. Springer, 2012, pp. 441–470.

[12] B. Sklar, "Rayleigh fading channels in mobile digital communication systems. i. characterization," *IEEE Communications magazine*, vol. 35, no. 7, pp. 90–100, 1997.

[13] S. Liang, H. Wan, T. Qin, J. Li, and W. Chen, "Multi-user computation offloading for mobile edge computing: A deep reinforcement learning and game theory approach," in *2020 IEEE 20th International Conference on Communication Technology (ICCT)*. IEEE, 2020, pp. 1534–1539.

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