Hierarchical Multi-Agent Deep Reinforcement Learning for Backscatter-aided Data Offloading

Hang Zhou*, Yusi Long*, Wenjie Zhang†, Jing Xu‡, and Shimin Gong*
*School of Intelligent Systems Engineering, Shenzhen Campus of Sun Yat-sen University, China
†School of Computer Sciences, Minnan Normal University, China
‡School of Electronic Information and Communications, Huazhong University of Science and Technology, China

2022 IEEE Wireless Communications and Networking Conference (WCNC) | 978-1-6654-4266-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/WCNC51071.2022.9771990

***Abstract*—In this paper, we consider a hybrid computation offloading scheme that allows edge users to offload workloads to the edge servers by using active RF communications and backscatter communications. We aim to maximize the overall energy efficiency by jointly optimizing the beamforming of access point (AP) and the users' offloading decisions. Considering a dynamic environment, we propose a hierarchical multi-agent deep reinforcement learning (H-MADRL) framework to solve this problem. The high-level agent resides in the AP and optimizes the beamforming strategy, while the low-level user agents learn and adapt individuals' offloading strategies. To further improve the learning efficiency, we propose a novel optimization-driven learning algorithm that allows the AP to estimate the low-level users' actions by solving an approximate problem efficiently. Then, the action estimation can be shared with all users and drive them to update individuals' actions independently. Simulation results reveal that our algorithm can improve the reward performance by 50%. The learning efficiency and reliability are also enhanced comparing to the conventional model-free learning methods.**

I. Introduction

Mobile edge computing (MEC) is recently proposed as a promising technology to alleviate the need for extensive computation at such IoT devices [1]. By deploying resource-rich MEC servers closer to the wireless IoT devices, the MEC servers' computation capabilities can be shared with all IoT devices by scheduling the IoT devices to offload their data and computation workloads to the MEC servers. Conventionally, data offloading via RF communications is inherently power-consuming, which may prevent the energy-limited edge users from using the MEC service. Thus, for energy-limited IoT devices, a more energy-efficient way will be required for data offloading to explore the potential of MEC service.

Comparing to the active RF communications, the passive wireless backscatter communications can be of extremely low power consumption, e.g., [2] and [3]. However, as a cost it has a lower data rate and becomes more vulnerable to the fluctuations of channel conditions. In this paper, we focus on the MEC offloading strategy in a hybrid radio network, in which each edge user can offload its workload by either using the active RF communications or the backscatter communications depending on its energy and workload statuses [4] and [5]. We aim to maximize the system's energy efficiency by optimizing the users' transmission scheduling and workload allocation among local computation, active and passive offloading. The first challenge lies in that the users' workload and energy supply may be uncertain in a dynamic channel environment. The energy consumption for data offloading also depends on the time-varying channel conditions. Another challenge comes from the users' competition for the MEC servers' computation resources, which have to be jointly optimized with the transmission control of the MEC-enabled access point (AP). This typically requires to solve a high-dimensional control problem, which can be difficult by conventional optimization methods.

Recently, deep reinforcement learning (DRL) methods have been successfully applied as a more flexible and robust approach to adapt the MEC offloading decisions by continuously interacting with the network environment, e.g., [6], [7]. The DQN method was employed in [8] and [9] to select the best MEC server for data offloading and resource allocation to maximize the energy efficiency subject to the users' workload deadline constraints. The actor-critic deep deterministic policy gradient (DDPG) algorithm was also studied in [10] to minimize the average end-to-end delay by learning the optimal policy for content caching, computing offloading, and radio resources allocation. In multi-user scenarios, the multi-agent DRL (MADRL) framework has been proposed to optimize the MEC offloading strategies. The authors in [11] employed the multi-agent DDPG (MADDPG) method to minimize the overall energy consumption, subject to the latency requirements, by adapting the computation offloading and interference coordination strategies. However, as the number of edge users increases, the learning performances of the MADRL methods become inefficient and unstable due to the rapidly increasing state and action spaces.

In this paper, we aim to improve the learning efficiency and energy efficiency in a multi-user hybrid offloading system. The maximization of the overall energy efficiency requires a joint optimization of the AP's beamforming strategy, the users' energy and workload allocations among local computation, and offloading through passive and active transmissions, called passive and active offloading, respectively. We reformulate the energy efficiency maximization problem as a Markov decision process (MDP) and propose the hierarchical MADRL framework to solve it. The basic idea is to distribute the decision-making process to both the AP and edge users. The high-level agent optimizes the AP's beamforming strategy

This work of Shimin Gong was supported in part by the National Natural Science Foundation of China under Grant 61972434 and the Shenzhen Fundamental Research Program under Grant JCYJ20190807154009444. The work of Jing Xu was supported in part by the National Science Foundation of Hubei Province under Grant 2020CFB794. (Corresponding author: Shimin Gong)

while the low-level user agents update their energy and workload allocations among local computation, passive and active offloading. The low-level users' strategy updates also drive the AP to adapt its beamforming strategy to improve the overall system performance. We further propose the optimization-driven learning algorithm for the low-level user agents. An optimization module is used to estimate the users' actions by solving an approximate problem efficiently. Given the action estimation, each edge user updates its own offloading decision independently by the single-agent DDPG method. The simulation results reveal that the hierarchial learning framework can reduce the action and state spaces and thus improve the learning efficiency significantly. The overall reward performance is also improved by 50% comparing to the model-free learning methods.

II. System Model

We consider a wireless network consisting of a multi-antenna AP with M antennas, co-located with the MEC server as illustrated in Fig. 1, and N wireless powered edge users, denoted by the set $\mathcal{N} = \{1, 2, \ldots, N\}$. The system model can be extended to the case with multiple MEC servers or APs. Similar to [12], each edge user can harvest RF energy from the AP's beamforming signals to sustain its operation, e.g., data transmission and processing. Each edge user can offload a part of its data and computation workload to the MEC server. The data offloading can be performed by either the active RF communications or the passive backscatter communications. The processed data at both the MEC server and the local user can be merged together to assist the user's decision making. We assume that the MEC can send back the processed data instantly, which typically has a much smaller size [12]. The complex uplink and downlink channels between the AP and the i-th edge user are denoted by $\mathbf{h}_i \in \mathcal{C}^{M\times 1}$ and $\mathbf{g}_i \in \mathcal{C}^{M\times 1}$, respectively. Each channel is considered to be frequency-flat block fading, i.e., the channel coefficients are constant in one time frame and may change frame by frame.

A. Workload and Time Allocation

We consider a time division protocol for multiple edge users to offload their data or workload to the MEC server. The time frame T_o is divided into multiple time slots and each edge user is allocated with a time slot T_i for $i \in \mathcal{N}$. We require that the processing of each user's workload has to be completed before the end of each time slot. Let ℓ_i denote the overall workload of the i-th user at the beginning of its time slot T_i. The workload allocation among local computation, passive and active offloading, denoted as $\ell_{c,i}$, $\ell_{b,i}$, and $\ell_{a,i}$, respectively, should satisfy the following constraint:

$$\ell_{a,i} + \ell_{b,i} + \ell_{c,i} \geq \ell_i, \quad \forall\, i \in \mathcal{N}. \tag{1}$$

The energy consumption of local computation depends on individual user's processing capability and the size of local workload $\ell_{c,i}$ for $i \in \mathcal{N}$. We expect that the low-power IoT users will have a constant data processing rate, considering a fixed operating frequency of the local processor, i.e., the number of CPU cycles per second, and hence the energy consumption in local computation can be evaluated by $e_{c,i} = \phi_c \ell_{c,i}$, where the constant ϕ_c denotes the user's energy consumption by processing a unit workload.

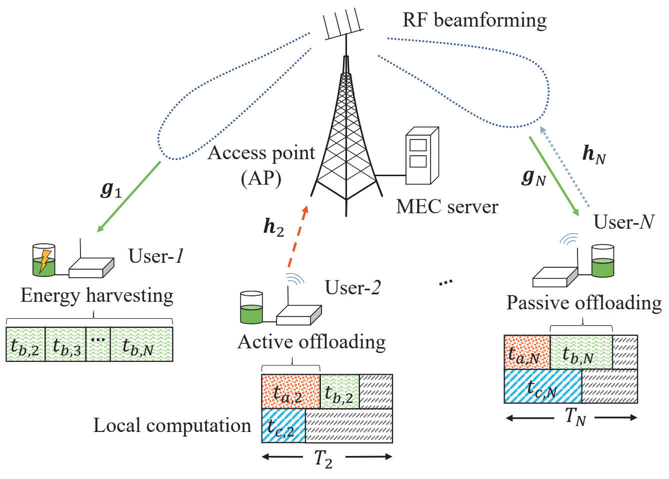


Fig. 1: Wireless powered hybrid computation offloading model. The time slots $(t_{a,i}, t_{b,i}, t_{c,i})$ denote the time allocations of the i-th user for active and passive offloading, as well as local computation.

We further divide each time slot T_i into two sub-slots, i.e., the active offloading sub-slot $t_{a,i}$ and the passive offloading sub-slot $t_{b,i}$, as illustrated in Fig 1. The user's workload must be processed successfully within each time slot:

$$t_{a,i} + t_{b,i} \leq T_i, \text{ and } t_{c,i} \leq T_i, \quad \forall\, i \in \mathcal{N}. \tag{2}$$

Note that the local workload can be processed in parallel with computation offloading and hence we require $t_{c,i} \leq T_i$. We assume that the time for MEC computation is much less than that for data offloading. Let $r_{a,i}$ and $r_{b,i}$ denote the data offloading rates in active and passive communications, respectively. The constraint in (1) can be rewritten as follows:

$$t_{a,i} r_{a,i} + t_{b,i} r_{b,i} + \ell_{c,i} \geq \ell_i, \quad \forall\, i \in \mathcal{N}. \tag{3}$$

This implies that the optimal workload or time allocation depends on the user's workload demand, energy supply, and data processing rates in different modes.

B. Hybrid Active and Passive Offloading

Let $\beta_{a,i}$ denote the transmit power of the i-th edge user. Then the received signal at the AP is given by $\mathbf{y}_{a,i} = \sqrt{\beta_{a,i}}\mathbf{h}_i x_i(t) + \mathbf{v}_o$, where $x_i(t)$ denotes the i-th edge user's information signal with unit power at time unit t. The vector $\mathbf{v}_o$ denotes the noise signals at AP. By maximal ratio combining (MRC), the active offloading rate is given by:

$$r_{a,i} = \log_2\left(1 + \beta_{a,i}|\mathbf{h}_i|^2\right), \tag{4}$$

where we assume a unit noise power for simplicity. Given the active offloading rate $r_{a,i}$, the power demand for active RF communications is determined by $\beta(r_{a,i}) \triangleq \left(2^{r_{a,i}} - 1\right)/|\mathbf{h}_i|^2$. Hence, the total power consumption in the active offloading can be estimated as $p_{a,i} = \beta\left(r_{a,i}\right) + p_o$, which includes $\beta(r_{a,i})$ for active communication and a constant p_o for powering the circuit. It is clear that $r_{a,i}$ depends on the channel conditions $\mathbf{h}_i$ and becomes stochastic in a dynamic channel environment.

In passive offloading, the AP emits carrier signals and simultaneously the edge user backscatters its workload information by reflecting the incident RF signals. Let $\mathbf{w}_i \in \mathcal{C}^{M\times 1}$ denote the normalized beamforming vector for the i-th edge user. The AP's RF beamforming in the i-th sub-slot is given by $\mathbf{u}_i(t) = \sqrt{p}\mathbf{w}_i s$, where p denotes the AP's transmit power and $s \in \mathcal{C}$ is a random symbol with unit power. The incident signal at the i-th edge user becomes $c_i(t) = \mathbf{h}_i^H \mathbf{u}_i(t)$ [1]. Meanwhile, the i-th user modulates its information $x_i(t)$ on the incident signal $c_i(t)$ by controlling the reflection coefficient $\Gamma(t) = \Gamma_o x_i(t)$, where Γ_o is an antenna-specific constant [13]. Then, the signal received by the AP is given by:

$$\mathbf{y}_{b,i}(t) = \Gamma_o \mathbf{h}_i^H x_i(t) c_i(t) + \mathbf{F}^H \mathbf{u}_i + \mathbf{v}_o, \tag{5}$$

where $\mathbf{F}$ represents the loop-back channel matrix and $\mathbf{v}_o$ denotes the noise signal. The first term in (5) contains the workload information $x_i(t)$ while the second term $\mathbf{F}^H\mathbf{u}_i$ is the self-interference due to the AP's signal beamforming, which can be filtered out by the AP assuming perfect interference cancellation. By using the MRC scheme, the received signal strength at the AP is given by $p|\Gamma_o|^2||\mathbf{h}_i||^2|\mathbf{h}_i^H\mathbf{w}_i|^2$, and thus we have the data rate in passive offloading as follows:

$$r_{b,i} = \log_2\left(1 + p|\Gamma_o|^2||\mathbf{h}_i||^2|\mathbf{h}_i^H\mathbf{w}_i|^2/|\mathbf{h}_i^H\mathbf{v}_o|^2\right). \tag{6}$$

III. Hierarchical Multi-agent Learning for Hybrid MEC Offloading

We aim to minimize the overall energy consumption of the MEC system, which includes two parts, i.e., the AP's RF beamforming and the MEC server's workload processing. The AP's energy consumption is given by $e_{b,i} = pt_{b,i}$ given the AP's transmit power p and the passive offloading time $t_{b,i}$ in each time slot, while the MEC server's energy consumption depends on the size of offloaded workload. By harvesting energy from the AP's beamforming signals, all edge users can sustain their local computation and data offloading.

A. Energy Minimization Problem

Let $p_{a,r}$ and $p_{b,r}$ (typically $p_{a,r} > p_{b,r}$) denote the AP's power demands in active and passive offloading, respectively. For each user-i, the energy consumption of the MEC system in data reception is given by $e_{r,i} \triangleq p_{a,r}t_{a,i} + p_{b,r}t_{b,i}$. Given the size of workload offloaded to the MEC server $\ell_{o,i} \triangleq \ell_{a,i} + \ell_{b,i}$, the energy consumption in workload computation can be evaluated by $e_{o,i} = \phi_o \ell_{o,i}$, where ϕ_o is a constant denoting the MEC server's energy efficiency. As such, the overall energy consumption is specified as follows:

$$E_o = \sum_{i\in\mathcal{N}} \Big(pt_{b,i} + p_{a,r}t_{a,i} + p_{b,r}t_{b,i} + \phi_o \ell_{o,i}\Big). \tag{7}$$

For each edge user, the active offloading relies on the energy stored in battery. Given the AP's beamforming signal $\mathbf{u}_i(t) = \sqrt{p}\mathbf{w}_i s$, the RF power harvested by the user-j in the i-th time slot can be approximated as $p_{j,i}^h \triangleq \eta\mathbb{E}[|\mathbf{h}_j^H\mathbf{u}_i(t)|^2] = \eta p|\mathbf{h}_j^H\mathbf{w}_i|^2$, where η denotes the energy harvesting efficiency [4]. Hence, the energy harvested by the user-j is given by:

$$E_j^h = \eta \sum_{i\in\mathcal{N}_{-j}} pt_{b,i}|\mathbf{h}_j^H\mathbf{w}_i|^2, \tag{8}$$

where $\mathcal{N}_{-j} = \mathcal{N}\setminus\{j\}$ denotes the set of all users excluding the j-th user. A non-linear energy harvesting model can also be considered here, similarly to that in [5] and [12]. To ensure the edge users' self-stainability, we require that the overall energy consumption is upper bounded as follows:

$$t_{a,j}p_{a,j} + \phi_c\ell_{c,j} \le \min\left\{E_j + E_j^h, E_{\max,j}\right\}, \quad \forall j \in \mathcal{N}, \tag{9}$$

where E_j denotes the initial energy status in battery and $E_{\max,j}$ denotes the maximum battery capacity of the j-th user. The energy consumption for passive offloading is omitted due to extremely low power consumption in backscatter communications. It is clear that each user's energy budget in (9) depends on the other users' time allocations.

We aim to maximize the overall energy efficiency by jointly optimizing the AP's beamforming strategy $\mathbf{w}_i$ and the edge users' hybrid offloading decisions, including the time allocation $\mathbf{t}_i \triangleq (t_{a,i}, t_{b,i}, t_{c,i})$ and workload division $\boldsymbol{\ell}_i \triangleq (\ell_{a,i}, \ell_{b,i}, \ell_{c,i})$ strategies for each user $i \in \mathcal{N}$, i.e.,

$$\min_{\mathbf{w}_i, \mathbf{t}_i, \boldsymbol{\ell}_i} \sum_{i\in\mathcal{N}} \Big(e_{b,i} + e_{r,i} + e_{o,i}\Big), \quad \text{s.t. (2), (3), and (9).} \tag{10}$$

The objective in problem (10) is linear in terms of the time allocation $\mathbf{t}_i$. The constraints in (2) and (3) represent the edge users' time and workload allocation constraints. The energy budget constraint in (9) brings the users' coupling via the workload division $\boldsymbol{\ell}_i$ and the time allocation $\mathbf{t}_i$. Another difficulty to solve (10) lies in the uncertain and dynamic network environment. The channel dynamics result in time-varying offloading rates while an underestimation of the workload may easily lead to the workload outage event, i.e., the users' actual workloads fail to be processed successfully within each time slot. These difficulties motivate us to design a robust learning approach that can adapt the AP's beamforming and the users' offloading decisions in a dynamic network environment.

B. High-level DDPG for Beamforming Optimization

In this part, we propose a hierarchical learning framework to improve the learning performance, motivated by a decomposition of the original optimization problem (10). The basic idea is to distribute the decision-making process to both the AP and multiple edge users. The AP first determines the high-level action, i.e., the beamforming strategy $\mathbf{w}_i$ for each user $i \in \mathcal{N}$, by using the conventional DDPG algorithm. Then, each low-level user $i \in \mathcal{N}$ updates its time and workload allocation strategy $(\mathbf{t}_i, \boldsymbol{\ell}_i)$ following the MADDPG algorithm. The low-level users' behaviors also drive the AP to update its beamforming strategy in the next decision epoch. Such a hierarchical design can reduce each agent's search space and potentially improve the learning efficiency.

[1]By channel reciprocity, here we assume that the uplink and downlink channels are the same $\mathbf{h}_i$ for each edge user.

The high-level system state $\mathbf{s}^o$ includes each user's workload demand ℓ_i, the energy status E_i, and the channel conditions between AP and the edge users $(\mathbf{h}_i, \mathbf{g}_i)$. The low-level users can report their local observations to the AP when they achieve a stable point. The high-level action $\mathbf{a}^o \triangleq [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N]$ includes the AP's beamforming strategy $\mathbf{w}_i$ for each user $i \in \mathcal{N}$. Given the system state $\mathbf{s}^o$, the high-level agent can adjust $\mathbf{a}^o$ to meet each user's offloading rate and the energy budget constraint. The high-level reward function $r_t^o(\mathbf{s}_t^o, \mathbf{a}_t^o)$ evaluates the overall energy efficiency based on the action $\mathbf{a}^o$ taken on the state $\mathbf{s}_o$. As indicated in the objective function of (10), the AP's reward function should be inversely proportional to the total energy consumption, and an increasing function of the successfully processed workload. Besides, the computation and energy resources will be wasted if the workload outage happens. Hence, we define the reward function as the product of the outage probability and the energy efficiency:

$$r_{\mathrm{t}}(\mathbf{s}_{\mathrm{t}}, \mathbf{a}_{\mathrm{t}}) = \left(\frac{1}{N} \sum_{i \in \mathcal{N}} x_i \right)^{\sigma} \left(\frac{1}{E_o} \sum_{i \in \mathcal{N}} x_i \ell_i \right), \qquad (11)$$

where E_o denotes the overall energy consumption in (7). The binary variable $x_i \in \{0, 1\}$ indicates whether workload outage happens or not, depending on the feasibility of the constraints in (10). Once workload outage happens, i.e., $x_i = 0$, the user's workload will be reprocessed in another time slot, which causes extra energy consumption and a wastage of computation resource. Hence, the first term $\frac{1}{N} \sum_{i \in \mathcal{N}} x_i$ in (11) can be viewed as the reliability of the system. A more reliable MEC system implies a smaller workload outage probability and that more workload can be processed successfully to improve the energy utilization. The second term $\frac{1}{E_o} \sum_{i \in \mathcal{N}} x_i \ell_i$ denotes the energy efficiency of the system, which is evaluated by the ratio between the successfully processed workload and the system's overall energy consumption, similar to that in [12]. The constant coefficient $\sigma > 0$ represents a tradeoff between reliability and efficiency. A larger σ implies that the MEC system is more sensitive to workload outage.

C. Low-level MADDPG for Time and Workload Allocation

Given the AP's beamforming strategy $\mathbf{a}^o$, each low-level user agent optimizes the offloading decision $(\mathbf{t}_i, \boldsymbol{\ell}_i)_{i \in \mathcal{N}}$ following the MADDPG algorithm. Specifically, each user $i \in \mathcal{N}$ can be viewed as an independent agent, adapting its offloading decision to improve its own reward based on its local observation of system. The system state of the i-th user, denoted as $\mathbf{s}_i^u \triangleq (E_i, \ell_i, \mathbf{h}_i, \mathbf{g}_i)$, includes its energy status E_i, workload demand ℓ_i, and the channel conditions $(\mathbf{h}_i, \mathbf{g}_i)$. The energy budget E_i is harvested from the AP's signal beamforming. The i-th user's action $\mathbf{a}_i^u \triangleq (\mathbf{t}_i, \boldsymbol{\ell}_i)$ is defined as the time and workload allocation among the local computing, active and passive offloading. Each edge user-i will get a higher reward if its workload can be successfully processed within the time slot T_i. Besides, a service price will be incurred if the edge user requests the MEC server's computation resources. Hence, we define the i-th edge user's reward function as follows:

$$r_i^u \triangleq \max\{x_i - \mu_1 \ell_{o,i}/\ell_i - \mu_2 t_{b,i}/T_i, 0\}, \qquad (12)$$

where $x_i \in \{0, 1\}$ denotes whether workload outage happens or not, similarly to that in (11). The constants μ_1 and μ_2 denote the unit resource prices for the MEC server and the AP, respectively. In particular, the first cost term $\mu_1 \ell_{o,i}/\ell_i$ represents the service cost that is proportional to the workload $\ell_{o,i}$ processed by the MEC server. The AP also consumes its energy by beamforming RF signals for passive offloading. This incurs the second cost term $\mu_2 t_{b,i}/T_i$ in (12), which is proportional to the passive offloading time $t_{b,i}$.

The low-level users can update their time and workload allocation strategies by the MADDPG algorithm. Each user agent's observation of the system not only depends on its own action, but also relates to the other agents' actions. As such, the value function of the i-th user can be revised as follows:

$$V^i_{\pi_i, \boldsymbol{\pi}_{-i}}(\mathbf{S}^u) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_i^u \Big(\mathbf{S}^u(t), \mathbf{A}^u(t) \Big) \right],$$

where π_i denotes the policy of the i-th user agent and $\boldsymbol{\pi}_{-i} \triangleq \{\pi_j\}_{j \neq i}$ represents the policies of all other user agents. $\mathbf{S}^u = [\mathbf{s}_1^u, \mathbf{s}_2^u, ..., \mathbf{s}_N^u]$ and $\mathbf{A}^u = [\mathbf{a}_1^u, \mathbf{a}_2^u, ..., \mathbf{a}_N^u]$ are the joint states and actions of all user agents. Each user's action is determined by its own policy, i.e, $\mathbf{a}_i^u(t) = \pi_i(\mathbf{s}_i^u(t))$. The interactions among low-level user agents can be processed by the centralized training and decentralized execution scheme [14]. During the offline training phase, the network controller, e.g., the AP or MEC server, collects all users' information and trains individual users' critic- and actor-networks simultaneously in a centralized manner. The well-trained critic- and actor-networks can be disseminated to the corresponding users and then used for online execution in a decentralized manner, i.e., each user can make its offloading decision according to its own actor-network.

D. Improving MADDPG via Action Estimation

Next, we try to improve the learning performance of the low-level MADDPG by exploiting the efficiency of model-based optimization methods and the robustness of model-free DDPG method. Given the AP's beamforming strategy, the high-level agent can also build an approximate model to estimate the user agents' actions, i.e., the offloading time and workload allocations, by solving an optimization problem with the fixed beamforming strategy at the AP. Then, the action estimation can be distributed to all low-level user agents and used to guide them to learn better rewards.

The AP's action estimation relies on the solution to the energy minimization problem in (10). The main difficulty lies in that the time and workload allocations $(t_{a,i}, \ell_{a,i})$ are non-convexly coupled in the constraint (9). However, by a similar approach as that in [15], we can easily verify that the problem (10) can be reformulated into a convex form. Hence, we can construct an optimization module to estimate each user agent's offloading decision efficiently. We further allow each user to improve its own action based on the AP's action estimation. Specifically, for each user agent $i \in \mathcal{N}$, it can use the conventional DDPG algorithm to update its own action $\mathbf{a}_i^u$ iteratively, assuming that all other user agents follow the AP's action estimation $\mathbf{a}_{-i}^u$. This can reduce the state and action

Algorithm 1 Optimization-driven H-MADDPG for Hybrid MEC Offloading

```
1:  Initialize DNNs and experience replay buffers of high-level and low-level agents
2:  **High-level AP's decision making**
3:  Update beamforming action $\mathbf{a}^o$ by DDPG algorithm
4:  Estimate low-level users' action $\mathbf{a}^u$ and the target value $y^*$ by solving problem (10)
5:  Distribute $(\mathbf{a}^o, \mathbf{a}^u)$ to all low-level user agents
6:  **Low-level user's decision making**
7:  **for** each low-level user agent $i \in \mathcal{N}$
8:      Fix other agents' action estimation $\mathbf{a}^u_{-i}$
9:      Update its own action $\tilde{\mathbf{a}}^u_i$ by DDPG algorithm
10:     Execute $\tilde{\mathbf{a}}^u_i$, observe reward, and buffer transitions
11:     Sample a mini-batch from the memory replay buffer
12:     Update the $i$-th critic and actor networks
13: **end for**
14: User agents report actions $\tilde{\mathbf{a}}^u = [\tilde{\mathbf{a}}^u_1, \tilde{\mathbf{a}}^u_2, \ldots, \tilde{\mathbf{a}}^u_N]$
15: AP estimates the target $y$ of the joint action $(\mathbf{a}^o, \tilde{\mathbf{a}}^u)$
16: **if** $y$ is greater than the optimization-driven target $y^*$
17:     AP updates the actions $(\mathbf{a}^o, \tilde{\mathbf{a}}^u)$ with prob. $1-\epsilon$ **else**
18:     AP updates the actions $(\mathbf{a}^o, \mathbf{a}^u)$ with prob. $1-\epsilon$
19: **end if**
20: Execute joint action, observe reward, and buffer transitions
21: Sample a mini-batch from the AP's replay buffer
22: Update the AP's critic and actor networks
23: Loop back to line (3)
```

spaces, and thus improve the learning efficiency of low-level user agents. The hierarchical learning framework also avoids frequent information exchange among low-level user agents and allows them to update individuals' actions independently based on local observations of the system. When all user agents find their optimal actions $\tilde{\mathbf{a}}^u_i$, they can report the actions to the AP and help update the AP's action estimation.

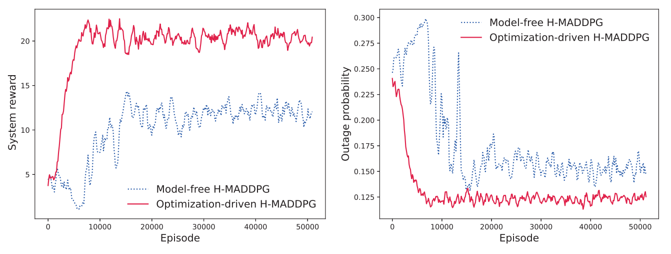
The optimization-driven hierarchical MADDPG (denoted as H-MADDPG) is detailed in Algorithm 1. It includes the outer-loop DDPG for the high-level AP and the inner-loop optimization-driven MADDPG for the low-level users. In lines $3-5$ of Algorithm 1, the AP chooses its beamforming strategy $\mathbf{a}^o$ by the DDPG algorithm and then estimates the users' time and workload allocation strategies $\mathbf{a}^u$. The action estimation is then informed to all users and used to assist their action explorations, as detailed in lines $7-13$ of Algorithm 1. By solving the model-based optimization (10), the AP can also evaluate the objective to estimate the target value $y*$, which can be viewed as the performance baseline during the learning process. Once the low-level users update their actions, the AP collects the users' actions $\tilde{\mathbf{a}}^u = [\tilde{\mathbf{a}}^u_1, \tilde{\mathbf{a}}^u_2, \ldots, \tilde{\mathbf{a}}^u_N]$ and updates the target value y in lines $14-15$ of Algorithm 1. A comparison between y and the optimization-driven target y^* will guide the AP's action update and transition to the next state, as shown in lines $16-22$ of Algorithm 1.

IV. Numerical Evaluation

In this section, we evaluate the performance improvement of the optimization-driven H-MADDPG framework with different parameters. We consider a specific topology, in which the MEC-enabled AP with $M = 3$ antennas is located in the origin of the coordinate. The locations of $N = 3$ edge users are given by $(5, 5)$, $(3, 3)$, and $(-2, 4)$, respectively. Each user is allocated a unit time slot. The energy harvesting efficiency is set as $\eta = 0.6$. The MEC service prices are set to $\mu_1 = 0.2$ and $\mu_2 = 0.4$. The random workload arrival at each edge user has a mean value $\lambda = 40$. The channels are assumed to be complex Gaussian distributions with zero mean and unit variance. The path loss follows a log-distance model $L_i = 20 + 20\log(d_i)$ (dB), where d_i denotes the distance from the AP to the i-th user. The parameter settings of each DDPG actor-critic network are listed in Table I, where `fc(m,n)` denotes a fully connected neural network with the size of $m \times n$.

TABLE I: Parameter settings in the H-MADDPG algorithm

Component	Network structure	Hyperparameter	Value
Actor	`fc(STATE_DIM,128), sigmoid`	`learning rate`	0.001
	`fc(128,64), sigmoid`	`batch size`	64
	`fc(64,ACTION_DIM), sigmoid`	`memory capacity`	2000
Critic	`fc(STATE_DIM,128)`	`reward discount`	0.5
	`fc(ACTION_DIM,128)`	`smoothing parameter`	0.1
	`relu, fc(128,1)`	ϵ`-greedy prob.`	0.5

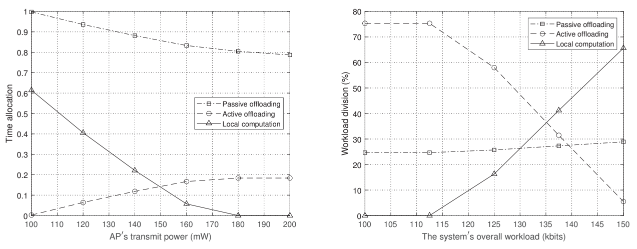


(a) Higher reward improvement **(b)** Lower outage performance

Fig. 2: The reward and outage in H-MADDPG algorithms.

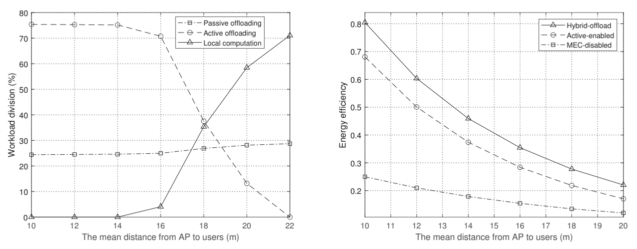
We first evaluate the convergence and learning performance of the optimization-driven H-MADDPG, comparing to the model-free H-MADDPG, which employs the classic MADDPG for low-level user agents. As shown in Fig. 2(a), the optimization-driven H-MADDPG converges faster and receives a significantly higher reward than that of the model-free H-MADDPG. Though the rewards fluctuate in both cases, we can expect 50% reward improvement by using the optimization-driven H-MADDPG. In Fig. 2(b), we observe that the optimization-driven H-MADDPG has a much lower outage probability than that of the model-free H-MADDPG. An outage event happens when the user's workload cannot be successfully processed at the end of its time slot. Thus, a lower outage probability indicates the enhanced reliability and robustness against dynamic network conditions. Besides, the steep decreasing of the outage probability in the early stage of learning also verifies a faster learning performance of the optimization-driven H-MADDPG.

We further examine the users' workload division between different offloading modes. When the AP's transmit power is low, the edge user has very limited energy budget and thus it will suppress the active offloading and prefer more workload division in passive offloading, as shown in Fig. 3(a). This ensures more energy harvesting opportunities for the edge



(a) Time allocation **(b)** Workload division

Fig. 3: Time and workload allocation in different computing schemes.



(a) Workload division **(b)** Energy efficiency

Fig. 4: The change of workload allocation and energy efficiency as the channel becomes worse off.

users. Besides, to meet the edge user's workload demand, we observe that a large part of the workload is processed by local computation. We also evaluate how the workload division changes with the overall workload of the system. As shown in Fig. 3(b), when the workload is small, the edge user has sufficient energy to offload all its workload to the MEC server by active and passive offloading. As the overall workload increases, the edge users' energy demands also increase. As such, the edge users need to perform more passive offloading to supply each other more energy, and correspondingly the time slot for active offloading is also reduced. This leads to the increase in local computation to meet the increasing workload demands, as shown in Fig. 3(b).

By gradually increasing the distances from the AP to edge users, we can evaluate the cases with deteriorating channel conditions. As the channels become worse, the edge users demand more energy to sustain the same data rate in active offloading. Hence, as shown in Fig. 4(a), a larger portion of the workload will be processed locally to save energy in active offloading. The remaining workload will be offloaded to the MEC server via passive offloading. Fig. 4(b) shows the change energy efficiency as the AP moves away from the edge users. We compare our scheme (denoted as the Hybrid-offload) with some baseline schemes. In the first baseline, the edge users only perform local computation by using the energy harvested from the AP, namely, the MEC-disabled scheme. The second baseline (denoted as the Active-enabled scheme) allows each user to perform local computation and offload a part of its workload to the MEC server by active RF communications. When the distances from the AP to the users increase, the AP has to consume more energy to maintain the same MEC services to all users. This implies the decreasing energy efficiency as shown in Fig. 4(b). As the channel becomes worse off, either active or passive offloading becomes costly. The user prefers to process its workload locally, which corroborates the observation in Fig. 4(a). When local computation dominates the workload, the energy efficiencies of the Hybrid-offload and Active-enabled schemes become closer to that of the MEC-disabled scheme, as shown in Fig. 4(b).

V. Conclusions

The energy minimization in a hybrid MEC system has been addressed by a hierarchical learning framework, consisting of a high-level agent and multiple low-level user agents. An optimization-driven learning algorithm has been devised to improve the learning efficiency. Numerical results have demonstrated that the proposed algorithm can improve the learning efficiency, reliability, and achieves a higher reward performance comparing to the model-free learning method.

References

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

[2] X. Lu, D. Niyato, H. Jiang, D. I. Kim, Y. Xiao, and Z. Han, "Ambient backscatter assisted wireless powered communications," *IEEE Wireless Commun.*, vol. 25, no. 2, pp. 170–177, Apr. 2018.

[3] G. Yang, D. Yuan, Y.-C. Liang, R. Zhang, and V. C. M. Leung, "Optimal resource allocation in full-duplex ambient backscatter communication networks for wireless-powered IoT," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2612–2625, Apr. 2019.

[4] P. X. Nguyen, D.-H. Tran, O. Onireti, P. T. Tin, S. Q. Nguyen, S. Chatzinotas, and H. Vincent Poor, "Backscatter-assisted data offloading in OFDMA-based wireless-powered mobile edge computing for IoT networks," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 9233–9243, Jun. 2021.

[5] F. Wang and X. Zhang, "Joint optimization for traffic-offloading and resource-allocation over RF-powered backscatter wireless networks," *IEEE J. Sel. Topic. Signal Process.*, Aug. 2021.

[6] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y. C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surv. Tut.*, vol. 21, no. 4, pp. 3133–3174, May 2019.

[7] S. Gong, Y. Xie, J. Xu, D. Niyato, and Y.-C. Liang, "Deep reinforcement learning for backscatter-aided data offloading in mobile edge computing," *IEEE Network*, vol. 34, no. 5, pp. 106–113, Sep. 2020.

[8] L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu, and L. Li, "Delay-aware and energy-efficient computation offloading in mobile edge computing using deep reinforcement learning," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 3, pp. 881–892, Mar. 2021.

[9] 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. Techn.*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.

[10] Y. Wei, F. R. Yu, M. Song, and Z. Han, "Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor-critic deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2061–2073, Apr. 2019.

[11] X. Huang, S. Leng, S. Maharjan, and Y. Zhang, "Multi-agent deep reinforcement learning for computation offloading and interference coordination in small cell networks," *IEEE Trans. Veh. Techn.*, Jul. 2021.

[12] F. Zhou and R. Q. Hu, "Computation efficiency maximization in wireless-powered mobile edge computing networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3170–3184, May 2020.

[13] S. Gong, L. Gao, J. Xu, Y. Guo, D. T. Hoang, and D. Niyato, "Exploiting backscatter-aided relay communications with hybrid access model in device-to-device networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 4, pp. 835–848, Dec. 2019.

[14] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. 31st Int. Conf. Neural Inf. Process. Sys. (NIPS'17)*, Dec. 2017, pp. 6382–6393.

[15] Y. Zou, J. Xu, S. Gong, Y. Guo, D. Niyato, and W. Cheng, "Backscatter-aided hybrid data offloading for wireless powered edge sensor networks," in *Proc. IEEE GLOBECOM*, Dec. 2019.