Towards Dynamic Energy/Carbon Trading and Resource Allocation for MEC: A Two-Timescale Deep Reinforcement Learning Approach

Xiaojing Chen[1], Yijun Ding[1], Wei Ni[2], Xin Wang[3], Yichuang Sun[4] and Shunqing Zhang[1]
[1]Key Laboratory of Specialty Fiber Optics and Optical Access Networks, Shanghai University, China
[2]Commonwealth Scientific and Industrial Research Organization (CSIRO), Australia
[3]Key Lab of EMW Information (MoE), Dept. of Commun. Sci. & Engr., Fudan University, China
[4]Department of Engineering and Technology, University of Hertfordshire, U.K.

***Abstract*—Integrating smart grid and carbon market into mobile edge computing (MEC) systems presents significant potential for reducing both operational energy costs and carbon footprints. This paper proposes a joint optimization framework for energy/carbon trading and resource allocation in MEC systems participating in grid-energy and carbon markets, aiming to minimize the long-term time-averaged cost of the energy/carbon tradings and the energy consumption of the system. Built on a two-timescale multi-agent deep reinforcement learning (TTMADRL) optimization framework, the Deep Deterministic Policy Gradient (DDPG) is generalized to make decisions on energy and carbon transactions at the large timescale; while at the small timescale, the task offloading schedules and CPU frequencies are distributively determined at each device by using the Multi-Agent DDPG (MADDPG) algorithm with enhanced scalability. Simulations demonstrate that the proposed TTMADRL achieves a 75.44% reduction in system costs compared to baseline approaches.**

***Index Terms*—Carbon market, deep reinforcement learning, mobile edge computing, smart grid, two timescales.**

I. Introduction

The evolution of the Internet of Things (IoT) gives rise to explosive demands for robust computing capabilities and low-latency communications. Mobile edge computing (MEC) emerges as a viable solution to this challenge by deploying edge servers in close proximity to IoT devices and enabling task execution at the network edge [1]. As the demand for MEC continues to expand, the associated energy expenses have become a substantial part of the operational expenditure for operators. The increasing carbon emission is also escalating into a crucial concern.

Smart grid technology significantly enhances the energy efficiency of MEC by incorporating features such as the integration of renewable energy sources (RES), bidirectional energy trading, and dynamic energy pricing [2]–[4]. Focusing on smart-grid-powered MEC, a few pioneering works have explored resource allocation and energy management to reduce the energy cost adapting to the dynamic environment [5], [6].

†Work in the paper was supported by the National Key R&D Program of China grants 2022YFB2902303 and 2022YFB2902304, Shanghai Rising-Star Program, and Shanghai Municipal Science and Technology Commission Foundation grant 24DP1500703.

On the other hand, carbon market has been launched with an expectation to reduce carbon emissions and raise revenues for operators through the transaction of carbon allowances. Only a few works has considered incorporating carbon trading mechanisms into MEC. Machine learning task offloading and carbon purchasing were jointly optimized to minimize the inference accuracy loss under a carbon purchasing budget in [7]. In a different yet relevant context, existing works have studied carbon trading strategies for energy systems [8], [9].

The interoperability of MEC, smart grid and carbon market needs to be supported in different timescales, as the energy/carbon pricing, energy harvesting, task demands and wireless channels generally change in different timescales. In this paper, we propose a novel two-timescale energy/carbon trading and resource allocation scheme for MEC systems engaging in integrated grid-energy and carbon markets. We aim to minimize the long-term time-averaged cost of the energy trading and carbon trading of the base station (BS), and the energy consumptions of the BS and the end devices. Built on the proposed two-timescale multi-agent deep reinforcement learning (TTMADRL) optimization framework, the edge server makes energy and carbon transactions by generalizing the Deep Deterministic Policy Gradient (DDPG) algorithm at the large timescale; while operating at the small timescale, the task offloading schedules (including task processing amount, offloading ratio and latency) and the CPU frequencies are distributively determined at each device using the Multi-Agent DDPG (MADDPG) algorithm, which enables distributed inference to reduce signaling overhead and ensure scalable and reliable performance. Simulations show that TTMADRL can save the system cost by 75.44%, compared to its benchmarks.

The rest of the paper is organized as follows. Section II describes the system model. Section III provides the problem formulation. Section IV developes the proposed TTMADRL algorithm. Section V presents the simulation results, followed by concluding remarks in Section VI.

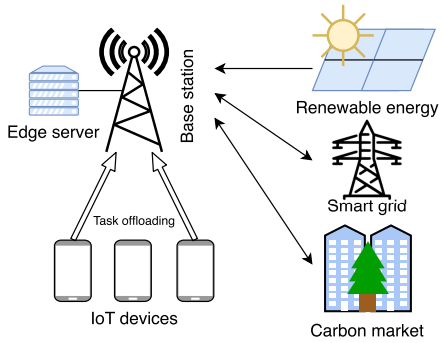

Fig. 1. Smart-grid-powered MEC engaging in integrated grid-energy and carbon markets.

II. System Models

Consider a smart-grid-powered MEC system, where a BS serves a set $\mathcal{I} := \{1, \ldots, I\}$ of IoT devices, as shown in Fig. 1. The BS is equipped with an edge server. Aside from the conventional energy supplied by the main grid, the BS can also harvest RES through, e.g., solar panels and/or wind turbines. It is installed with a rechargeable battery to store energy and prevent energy outages. Relying on the two-way energy trading facility, the BS can also buy energy from or sell energy to the main grid at dynamically changing market prices. The BS participates in carbon allowance trading through the carbon market. When the carbon emissions of the BS exceed its permitted amount, it must purchase additional carbon allowances to compensate for the excess emissions. Conversely, the BS can sell its surplus carbon allowances to other entities through the market. The BS can also engage in renewable energy trading, which is in accordance with the renewable energy certificate market [10]. Renewable energy certificates verify the amount of renewable energy produced. Consumers (i.e., the BSs) receive these certificates if they purchase renewable sources, certifying their contributions to carbon footprint reduction.

The computational tasks of IoT devices can be executed locally, offloaded to the edge server or processed through a hybrid way, supposing that each IoT device has a finite buffer to backlog unprocessed tasks following the first-in-first-out (FIFO) queuing discipline.

We adopt a two-timescale framework to model the system. The large time interval is indexed by $k \in \mathcal{K} := \{0, 1, \ldots, K-1\}$, where K is the number of the large time intervals. Each time interval comprises T time slots, concurring with the energy and carbon trading timescale. The small time slot is indicated by $t \in \mathcal{T} := \{0, 1, \ldots, KT-1\}$, which is no longer than the coherence time of channels. Hence, $k = \lfloor t/T \rfloor$. The length of each small time slot is τ, and the length of each large time interval is $T\tau$.

A. Task Queue Model

Let $R_{u,i}^t \in [0, R_{\max}]$ denote the newly arrived task size of device i at time slot t, where $R_{\max}$ is the maximum task arrival size. Let $R_{\text{pro},i}^t$ denote the task size of device i processed at time slot t, and $Q_{u,i}^t$ denote the queue length of device i at slot t. The system decides whether to offload part of the tasks to the edge server for processing, or to process them locally on the IoT devices. The queue length at slot $(t+1)$ is

$$Q_{u,i}^{t+1} = \min\{Q_{u,i}^t - R_{\text{pro},i}^t + R_{u,i}^t, Q_{\max}\}, \quad \forall i \in \mathcal{I}, t \in \mathcal{T}, \tag{1}$$

where $R_{\text{pro},i}^t \leq Q_{u,i}^t + R_{u,i}^t$ due to causality, and $Q_{\max}$ is the queue buffer size.

B. Task Execution and Energy Consumption

1) IoT Devices: Let $f_{u,i}^t \in [0, f_u^{\max}]$ denote the CPU frequency of device i at time slot t, where $f_u^{\max}$ is the maximum CPU frequency. The power consumed by device i for local execution at time slot t is given by $P_{\text{loc},i}^t = \epsilon_i (f_{u,i}^t)^3$, where ϵ_i is the capacitance coefficient.

Let $T_{\text{loc},i}^t$ denote the latency of task execution at device i at time slot t, as given by

$$T_{\text{loc},i}^t = \frac{a_{\text{ratio},i}^t R_{\text{pro},i}^t K}{f_{u,i}^t} \leq \tau, \quad \forall i \in \mathcal{I}, t \in \mathcal{T}, \tag{2}$$

where $a_{\text{ratio},i}^t \in [0,1]$ is the ratio of tasks processed at device i at slot t, and K is the number of CPU cycles required to process a nat of data. Thus, the energy consumption of local task computing at device i at time slot t is

$$E_{\text{loc},i}^t = P_{\text{loc},i}^t T_{\text{loc},i}^t = \epsilon_i (f_{u,i}^t)^3 T_{\text{loc},i}^t, \quad \forall i \in \mathcal{I}, t \in \mathcal{T}. \tag{3}$$

Each IoT device can offload part of its tasks to the edge server for remote execution. Based on the Shannon formula, the transmit power of device i for offloading at slot t is

$$P_{\text{off},i}^t = \frac{\sigma_t^2}{h_i^t} \left(e^{\frac{(1-a_i^t) R_{\text{pro},i}^t}{w_i^t T_{\text{off}}}} - 1\right), \quad \forall i \in \mathcal{I}, t \in \mathcal{T}, \tag{4}$$

where σ_t^2 is the noise power at the BS, h_i^t is the effective channel gain from device i to the BS, w_i^t is the bandwidth assigned to device i, and $T_{\text{off},i}^t$ is the latency for task offloading.

The energy consumption of device i for task offloading at time slot t is given by

$$E_{\text{off},i}^t = P_{\text{off},i}^t T_{\text{off},i}^t, \quad \forall i \in \mathcal{I}, t \in \mathcal{T}. \tag{5}$$

Therefore, the energy consumption of device i is given by $E_{u,i}^t = E_{\text{loc},i}^t + E_{\text{off},i}^t$.

2) BS: Let $f_{s,i}^t$ denote the CPU frequency of the edge server assigned to the task of device i at time slot t. $0 \leq \sum_i f_{s,i}^t \leq f_s^{\max}$, where $f_s^{\max}$ is the maximum CPU frequency of the edge server. The corresponding power consumption $P_{\text{edge},i}^t = \delta (f_{s,i}^t)^3$, where δ is the capacitance coefficient of the edge server. Let $T_{\text{com},i}^t$ denote the latency of remotely processing the task of device i at time slot t, as given by

$$T_{\text{com},i}^t = \frac{(1 - a_{\text{ratio},i}^t) R_{\text{pro},i}^t K}{f_{s,i}^t} \leq \tau - T_{\text{off},i}^t, \quad \forall i \in \mathcal{I}, t \in \mathcal{T}. \tag{6}$$

The total energy consumption of task execution at the edge server is

$$E_{\text{com},s}^t = \sum_i E_{\text{com},s,i}^t = \sum_i P_{\text{edge},i}^t T_{\text{com},i}^t, \quad \forall t \in \mathcal{T}. \tag{7}$$

The total energy consumption of the BS E_s^t accounts for the energy consumption of task execution $E_{\text{com},s}^t$, and a static energy consumption $E_{\text{static},s}$ incurred by, e.g., circuits, battery backup, and cooling. That is, $E_s^t = E_{\text{com},s}^t + E_{\text{static},s}$.

Let $E_{\text{bat},s}^t$ represent the battery charging amount of the BS at time slot t, where $E_{\text{bat},s}^t > 0$ indicates energy charging, and $E_{\text{bat},s}^t < 0$ indicates energy discharging. Given B_s^t as the battery energy level of the BS at the beginning of time slot t, the battery's energy dynamics can be expressed as

$$B_s^{t+1} = B_s^t + E_{\text{bat},s}^t,\ B^{\min} \le B_s^t \le B^{\max},\ \forall t \in \mathcal{T}, \quad (8)$$

where $B^{\min}$ is the minimum required energy reserved in the battery, and $B^{\max}$ is the battery size.

C. Energy and Carbon Allowances Trading

1) Energy Trading: Let $E_{\text{tra},s}^k$ denote the traditional energy traded at time interval k, and $E_{\text{res},s}^k$ denote the traded renewable energy. $E_{\text{tra},s}^k > 0$ or $E_{\text{res},s}^k > 0$ indicates the BS purchases traditional or renewable energy; and $E_{\text{tra},s}^k < 0$ or $E_{\text{res},s}^k < 0$ indicates the BS sells traditional or renewable energy.

Let α_{tra}^k and β_{tra}^k denote the buying and selling prices of traditional energy at time interval k, respectively. In practice, $\alpha_{\text{tra}}^k > \beta_{\text{tra}}^k > 0$. The traditional energy transaction cost of the BS at time interval k is $G(E_{\text{tra},s}^k) = \alpha_{\text{tra}}^k [E_{\text{tra},s}^k]^+ - \beta_{\text{tra}}^k [-E_{\text{tra},s}^k]^+$. Let α_{res}^k and β_{res}^k denote the buying and selling prices of renewable energy at time interval k, respectively. The renewable energy transaction cost of the BS at time interval k is $G(E_{\text{res},s}^k) = \alpha_{\text{res}}^k [E_{\text{res},s}^k]^+ - \beta_{\text{res}}^t [-E_{\text{res},s}^k]^+$. Let e_s^k denote the renewable energy generated at time interval k. The cost of renewable energy generation is $G(e_s^k) = \alpha_g e_s^k$, where α_g is the Levelized Cost of Electricity (LCOE) of renewable energy generation.

Consequently, the following holds per time interval k to balance the energy supply and demand:

$$\sum_{t=kT}^{(k+1)T-1} (E_{\text{bat},s}^t + E_s^t) = E_{\text{tra},s}^k + E_{\text{res},s}^k + e_s^k, \quad \forall k \in \mathcal{K}. \quad (9)$$

The carbon emissions of the system can be given by $C_{\text{car}}^k = \rho_{\text{grid}} E_{\text{tra},s}^k + \rho_g (e_s^k + E_{\text{res}}^k)$, where ρ_{grid} (in kgCO_2/kWh) is the carbon factor of conventional thermal energy generation, and ρ_g is the carbon factor of renewable energy generation, stemming from the carbon emissions of the manufacturing and recycling of renewable energy equipment. In practice, $\rho_{\text{grid}} > \rho_g$.

2) Carbon Allowances Trading: Let A^k denote the carbon emission allowances of the BS at the beginning of the large time interval k, and A_{tra}^k denote the carbon allowances traded in time interval k. Let $A^{\max}$ denote the maximum carbon allowance that the BS can have. It holds that

$$A^{k+1} = \min\{A^k - C_{\text{car}}^k + A_{\text{tra}}^k, A^{\max}\}, \quad \forall k \in \mathcal{K}, \quad (10)$$

where $C_{\text{car}}^k \le A^k + A_{\text{tra}}^k$ due to causality.

Let α_{car}^k and β_{car}^k denote the buying and selling prices of carbon allowances at time interval k, respectively. The transaction cost of carbon allowances at time interval k is given by $G(A_{\text{tra}}^k) = \alpha_{\text{car}}^k [A_{\text{tra}}^k]^+ - \beta_{\text{car}}^k [-A_{\text{tra}}^k]^+$.

The total cost of the BS at time interval k consists of the traditional energy trading cost, the renewable energy trading cost, the renewable energy generation cost, and the carbon allowance trading cost, as given by

$$\Phi_k = G(E_{\text{tra},s}^k) + G(E_{\text{res},s}^k) + G(e_s^k) + G(A_{\text{tra}}^k). \quad (11)$$

III. PROBLEM FORMULATION

We cast a two-timescale energy/carbon trading and resource allocation problem to minimize the time-averaged system cost. The system cost is defined as the weighted sum of the BS's cost Φ_k and the system energy consumption $\sum_{i\in\mathcal{I}} E_{u,i}^t + E_s^t$. To solve the problem, we need to jointly optimize the traditional energy trading $\{E_{\text{tra},s}^k, \forall k\}$, renewable energy trading $\{E_{\text{res},s}^k, \forall k\}$, carbon allowance trading $\{A_{\text{tra}}^k, \forall k\}$, CPU frequencies $\{f_{u,i}^t, f_{s,i}^t, \forall i, t\}$, task processing amount $\{R_{\text{pro},i}^t, \forall i, t\}$, task offloading ratio $\{a_{\text{ratio},i}^t, \forall i, t\}$, and task offloading latency $\{T_{\text{off},i}^t, \forall i, t\}$.

Let $\mathcal{X} = \{f_{u,i}^t, f_{s,i}^t, T_{\text{off},i}^t, R_{\text{pro},i}^t, a_{\text{ratio},i}^t, E_{\text{tra},s}^k, E_{\text{res},s}^k, A_{\text{tra}}^k \ \forall i, t, k\}$. The problem of interest is

$$\mathbf{P1}: \min_{\mathcal{X}} \lim_{K\to\infty} \frac{1}{K} \{ \sum_{k=0}^{K-1} \Phi_k + m \sum_{t=kT}^{(k+1)T-1} (\sum_{i=0}^{I-1} E_{u,i}^t + E_s^t) \} \quad (12a)$$

$$\text{s. t.} \quad R_{\text{pro},i}^t \le Q_{u,i}^t + R_{u,i}^t,\ C_{\text{car}}^k \le A^k + A_{\text{tra}}^k, \quad (12b)$$

$$0 \le f_{u,i}^t \le f_{u,i}^{\max}, \quad 0 \le \sum_i f_{s,i}^t \le f_s^{\max}, \quad (12c)$$

$$(1), (2), (6), (8), (9), (10).$$

where m is the weight.

Problem $\mathbf{P1}$ is a challenging nonlinear programming problem characterized by strongly coupled decision variables. Specifically, the task queue length dynamics in (1), the battery (dis)charging behavior in (8), and the evolution of carbon allowances in (10) couple the variables over time. Moreover, the energy/carbon trading decisions made at the large time interval affect the sequent resource allocation. Problem $\mathbf{P1}$ aims to minimize the long-term time-averaged system cost in the presence of temporal variations, which is impractical and computationally prohibitive.

We decompose Problem $\mathbf{P1}$ into two subproblems. At the large time intervals, we minimize the time-averaged cost of the BS, i.e.,

$$\mathbf{P2}: \min_{\boldsymbol{x}_{\text{large}}^k} \lim_{K\to\infty} \frac{1}{K} \sum_{k=0}^{K} \Phi_k$$
$$\text{s. t. } (8), (10), (12b).$$

where $\boldsymbol{x}_{\text{large}}^k := \{E_{\text{tra},s}^k, E_{\text{res},s}^k, A_{\text{tra}}^k\}$.

At each time slot, given the energy/carbon trading decisions, Problem $\mathbf{P1}$ is reduced to minimizing the energy consumption of the system, as given by

$$\mathbf{P3}: \min_{\boldsymbol{x}_{\text{small}}^t} \lim_{K\to\infty} \sum_{t=0}^{KT-1} (\sum_{i=0}^{I-1} E_{u,i}^t + E_s^t)$$
$$\text{s. t.} \quad (1), (2), (6), (8), (12b), (12c),$$

where $\boldsymbol{x}^t_{\text{small}} := \{f^t_{u,i}, f^t_{s,i}, T^t_{\text{off},i}, R^t_{\text{pro},i}, a^t_{\text{ratio},i}\}$.

In what follows, we establish the lemmas to reveal the properties of the optimal CPU frequencies at the devices and the edge server, respectively.

Lemma 1: The optimal CPU frequencies of the devices satisfy: $f^t_{u,i} = \frac{a^t_{\text{ratio},i} R^t_{\text{pro},i} K}{\tau} \leq f^{\max}_u, \ \forall i, t.$

Proof: It can be observed from (3) that a lower CPU frequency at the device results in a reduced energy consumption. By extending the local task execution latency $T^t_{\text{loc},i}$ in (2) to the duration of a time slot τ, the CPU frequency at the device is minimized, such that the energy consumption of local task computing at device is minimized. ■

Lemma 2: The CPU frequency of the edge server satisfies: $f^t_{s,i} = \frac{(1-a^t_{\text{ratio},i}) R^t_{\text{pro},i} K}{\tau - T^t_{\text{off},i}}, \ \forall i, t.$

Proof: This proof is similar to that of Lemma 1, and hence is omitted here. ■

IV. Proposed TTMADRL Framework

Noticing Problems **P2** and **P3** entail Markov decision processes (MDPs), we propose a novel DRL-based control algorithm, TTMADRL, to solve the problems at different timescales. TTMADRL is illustrated in Fig. 2 and summarized in Algorithm 1, which comprises two layers: the edge layer and the device layer. The energy trading and carbon allowance trading in Problem **P2** are optimized by the edge server using DDPG per large time interval, while the task offloading strategy in Problem **P3** is distributively determined by each device using MADDPG per small time slot. The two layers are trained alternately with their respective parameters sequentially updated to enhance the overall system performance.

A. Edge Layer

The DDPG agent deployed at the edge server incorporates two fundamental networks: the actor policy network approximating the optimal energy/carbon trading policies, and the critic network estimating the associated Q-value functions. To enhance the stability and robustness of the training process, target actor and critic networks are integrated and updated using a soft update mechanism to ensure incremental adjustments. Moreover, a replay buffer is utilized to store interaction experiences, facilitating randomized sampling for network training to enhance learning efficiency and reliability.

1) State: The state at the edge layer is defined as $\boldsymbol{S} = \{\boldsymbol{s}_k, \forall k\}$, where $\boldsymbol{s}_k := \{\alpha^k_{\text{tra}}, \alpha^k_{\text{res}}, \alpha^k_{\text{car}}, \beta^k_{\text{tra}}, \beta^k_{\text{res}}, \beta^k_{\text{car}}, B^{kT}_s, A^k, e^k_s\}$ collects the states at time interval k. The DDPG agent observes the system states and selects the optimal actions for energy trading and carbon trading.

2) Action: The action is defined as $\boldsymbol{A}_{\text{large}} = \{\boldsymbol{a}_k, \forall k\}$, where $\boldsymbol{a}_k := \{E^k_{\text{tra},s}, E^k_{\text{res},s}, A^k_{\text{tra}}\}$ collects the actions at time interval k.

3) Reward: A reward evaluates the action executed. The immediate reward is defined as $r_k := -\Phi_k$. The reward is set to a large negative value when the constraints in Problem **P2** are not satisfied.

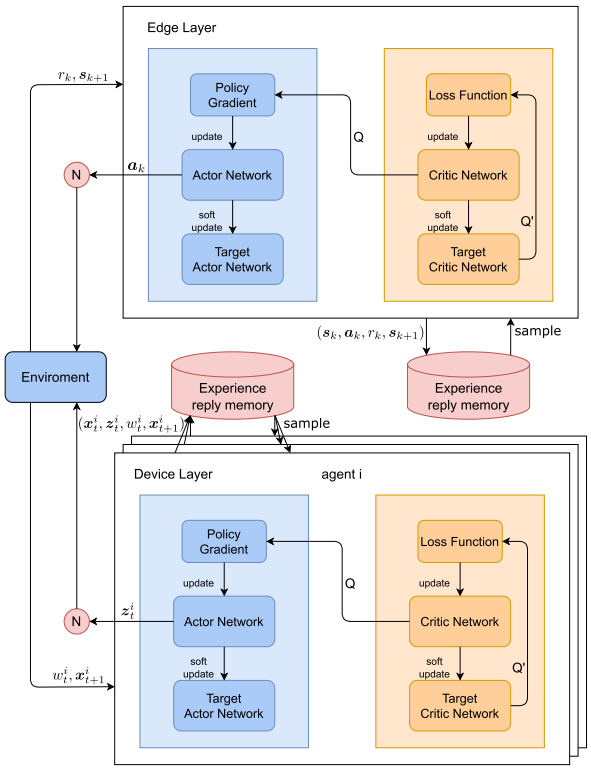

Fig. 2. The proposed TTMADRL framework comprising the edge layer and the device layer.

4) Network Update: At every time interval k, the parameters of the critic network, denoted as $\boldsymbol{\theta}^Q$, are updated by minimizing the following loss function:

$$\mathcal{L}(\boldsymbol{\theta}^Q) = \frac{1}{N} \sum_k \Big(y_k - Q_k(\boldsymbol{s}_k, \boldsymbol{a}_k | \boldsymbol{\theta}^Q) \Big)^2, \tag{15}$$

where N is the mini-batch size, $Q_k(\boldsymbol{s}_k, \boldsymbol{a}_k | \boldsymbol{\theta}^Q)$ is the Q-value predicted by the critic network; $y_k = r_k + \gamma_{\text{large}} Q'(\boldsymbol{s}_{k+1}, \mu'_k(\boldsymbol{s}_{k+1} | \boldsymbol{\theta}^{\mu'}) | \boldsymbol{\theta}^{Q'})$ is the target Q-value; γ_{large} is the discount factor; $\boldsymbol{\theta}^{\mu'}$ and $\boldsymbol{\theta}^{Q'}$ are the parameters of the target actor and critic networks, respectively; $\mu'_k(\boldsymbol{s}_{k+1} | \boldsymbol{\theta}^{\mu'})$ is the action predicted by the target actor network in state $\boldsymbol{s}_{k+1}$; $Q'(\boldsymbol{s}_{k+1}, \mu'_k(\boldsymbol{s}_{k+1} | \boldsymbol{\theta}^{\mu'}) | \boldsymbol{\theta}^{Q'})$ is the Q-value estimated by the target critic network for the next state $\boldsymbol{s}_{k+1}$.

The actor network optimizes the policy by directly maximizing the expected Q-value. The policy gradient used to update its parameters $\boldsymbol{\theta}^\mu$ is given by

$$\nabla_{\boldsymbol{\theta}^\mu} J \approx \frac{1}{N} \sum_k \nabla_{\boldsymbol{a}} Q_k(\boldsymbol{s}_k, \boldsymbol{a} | \boldsymbol{\theta}^Q) \big|_{\boldsymbol{a} = \mu_k(\boldsymbol{s}_k | \theta^\mu)} \nabla_{\boldsymbol{\theta}^\mu} \mu_k(\boldsymbol{s}_k | \boldsymbol{\theta}^\mu), \tag{16}$$

where $\nabla_a Q_k(\boldsymbol{s}_k, \boldsymbol{a} | \boldsymbol{\theta}^Q) \big|_{\boldsymbol{a} = \mu_k(\boldsymbol{s}_k | \theta^\mu)}$ is the gradient of the Q-value with respect to the action, evaluated at $\boldsymbol{s}_k$ and $\boldsymbol{a} = \mu_k(\boldsymbol{s}_k)$, and $\nabla_{\boldsymbol{\theta}^\mu} \mu_k(\boldsymbol{s}_k | \boldsymbol{\theta}^\mu)$ is the gradient of the actor's policy with respect to its parameters.

The target actor and critic networks are updated softly by

$$\boldsymbol{\theta}^{\mu'} \leftarrow \xi_{\text{large}} \boldsymbol{\theta}^\mu + (1 - \xi_{\text{large}}) \boldsymbol{\theta}^{\mu'}; \boldsymbol{\theta}^{Q'} \leftarrow \xi_{\text{large}} \boldsymbol{\theta}^Q + (1 - \xi_{\text{large}}) \boldsymbol{\theta}^{Q'}, \tag{17}$$

where $\xi_{\text{large}} \in (0, 1)$ is the soft update coefficient.

B. Device Layer

The MADDPG is employed at the device layer to optimize the task offloading strategy of devices. The structure of the MADDPG parallelizes that of the DDPG, with the key distinction lying in that by accommodating the agents at each device, the MADDPG is suitable to our scenario where the devices share and compete for radio and computational resources. During the training phase, the agents have centralized access to the observations and actions of all other agents. By contrast, during inference, they rely solely on their own observations, exhibiting a decentralized feature.

1) Observation: The observation at the device layer is defined as $\boldsymbol{X} = \{\boldsymbol{x}_t^i, \forall i, t\}$, where $\boldsymbol{x}_t^i := \{Q_{u,i}^t, h_i^t, R_{u,i}^t, B_s^t\}$ collects the observation of the agent deployed at device i at t.

2) Action: The action is defined as $\boldsymbol{Z}_{\text{small}} = \{\boldsymbol{z}_t^i, \forall i, t\}$, where $\boldsymbol{z}_t^i := \{R_{\text{pro},i}^t, a_{\text{ratio},i}^t, T_{\text{off},i}^t\}$ collects the actions of the agent deployed at device i at time slot t.

3) Reward: The immediate reward is defined as $w_t^i := -(E_{u,i}^t + E_{\text{com},s,i}^t)$. The reward is set to a large negative value when the constraints in Problem $\mathbf{P3}$ are not satisfied.

4) Network Update: The parameters of the critic network of agent i, denoted as $\boldsymbol{\psi}_i^Q$, are updated by minimizing the following loss function:

$$\mathcal{L}(\boldsymbol{\psi}_i^Q) = \frac{1}{S}\sum_t \left(y_t^i - Q_t^i(\boldsymbol{x}_t, \boldsymbol{z}_t|\boldsymbol{\psi}_i^Q)\right)^2, \tag{18}$$

where S is the minibatch size, $\boldsymbol{x}_t = \{\boldsymbol{x}_t^i, \forall i\}$, $\boldsymbol{z}_t = \{\boldsymbol{z}_t^i, \forall i\}$, $Q_t^i(\boldsymbol{x}_t, \boldsymbol{z}_t|\boldsymbol{\psi}_i^Q)$ is the Q-value estimated by the critic network of agent i, and y_t^i is the target Q-value of agents i, as given by

$$y_t^i = w_t^i + \gamma_{\text{small}} Q_t^{i'}(\boldsymbol{x}_{t+1}, \boldsymbol{z}_{t+1}|\boldsymbol{\psi}_i^{Q'})\Big|_{\boldsymbol{z}_{t+1}^i = \mu_t'(\boldsymbol{x}_{t+1}^i|\boldsymbol{\psi}_i^{\mu'})}, \tag{19}$$

where γ_{small} is the discount factor; $\boldsymbol{\psi}_i^{Q'}$ and $\boldsymbol{\psi}_i^{\mu'}$ are the parameters of the target critic and actor networks for agent i, respectively; $Q_t^{i'}(\boldsymbol{x}_{t+1}, \boldsymbol{z}_{t+1}|\boldsymbol{\psi}_i^{Q'})$ is the Q-value estimated by the target critic network of agent i; $\mu_t'(\boldsymbol{x}_{t+1}^i|\boldsymbol{\psi}_i^{\mu'})$ is the action predicted by the target actor network of agent i.

The parameters of the actor network $\boldsymbol{\psi}_i^\mu$ of agent i are updated using the sampled policy gradient:

$$\nabla_{\boldsymbol{\psi}_i^\mu} J \approx \frac{1}{S}\sum_t \nabla_{\boldsymbol{z}_t^i} Q_t^i(\boldsymbol{x}_t, \boldsymbol{z}_t|\boldsymbol{\psi}_i^Q)\Big|_{\boldsymbol{z}_t^i = \mu_t^i(\boldsymbol{x}_t^i|\boldsymbol{\psi}_i^\mu)} \nabla_{\boldsymbol{\psi}_i^\mu} \mu_t^i(\boldsymbol{x}_t^i|\boldsymbol{\psi}_i^\mu), \tag{20}$$

where $\mu_t^i(\boldsymbol{x}_t^i|\boldsymbol{\psi}_i^\mu)$ is the action predicted by the actor network of agent i.

The target networks are updated softly by

$$\boldsymbol{\psi}_i^{\mu'} \leftarrow \xi_{\text{small}}\boldsymbol{\psi}_i^\mu + (1-\xi_{\text{small}})\boldsymbol{\psi}_i^{\mu'}; \boldsymbol{\psi}_i^{Q'} \leftarrow \xi_{\text{small}}\boldsymbol{\psi}_i^Q + (1-\xi_{\text{small}})\boldsymbol{\psi}_i^{Q'}, \tag{21}$$

where $\xi_{\text{small}} \in (0, 1)$ is the soft update coefficient.

Algorithm 1: Proposed TTMADRL Algorithm

Initialize the DDPG and MADDPG networks with random model parameters. Initialize the replay buffers.
for episode = 0 to $M-1$ **do**
  %%% EDGE LAYER
  Receive the initial state $\boldsymbol{s}_1$.
  for k in 0 to $K-1$ **do**
    Select $\boldsymbol{a}_k = \mu_k(\boldsymbol{s}_k|\boldsymbol{\theta}^\mu) + \mathcal{N}$ via the actor network with the expoloration noise $\mathcal{N}$.
    Execute $\boldsymbol{a}_k$, (dis)charge the battery via (9), and update the carbon allowance by (10).
    Obtain reward r_k and the subsequent state $\boldsymbol{s}_{k+1}$.
    Store $(\boldsymbol{s}_k, \boldsymbol{a}_k, r_k, \boldsymbol{s}_{k+1})$ in the replay buffer R.
    if the replay buffer size is greater than the batch size **then**
      Sample a random mini-batch of N transitions $(\boldsymbol{s}_k, \boldsymbol{a}_k, r_k, \boldsymbol{s}_{k+1})$ from R, and update the critic and actor networks by (15) and (16), respectively.
      Update the target networks via (17).
    end if
    %%% DEVICE LAYER
    Receive the initial state $\boldsymbol{x}_{kT}$.
    for t in kT to $(k+1)T-1$ **do**
      For each agent i, select $\boldsymbol{z}_t^i = \mu_t^i(\boldsymbol{x}_t^i|\boldsymbol{\psi}_i^\mu) + \mathcal{N}'$ via the actor network with the exploration noise $\mathcal{N}'$.
      Execute $\boldsymbol{z}_t^i$, (dis)charge the battery via (9), and update the task queue by (1).
      Obtain reward w_t^i and the subsequent state $\boldsymbol{x}_{t+1}^i$.
      Store $(\boldsymbol{x}_t^i, \boldsymbol{z}_t^i, w_t^i, \boldsymbol{x}_{t+1}^i)$ in the replay buffer D.
      if the replay buffer size is greater than the batch size **then**
        Sample a batch from the replay buffer D.
        Update the critic and actor networks by (18) and (20), respectively.
        Update the target networks via (21).
      end if
    end for
  end for
end for

V. Simulation Results

We set $I = 5$, $Q_{\max} = 8\times10^6$ nats, $f_u^{\max} = f_s^{\max} = 1$ GHz, $\epsilon_i = \delta = 10^{-27}, \sigma_t^2 = 10^{-9}, W = 1$ MHz, $P_{\text{static},s} = 5$ KWh, $B^{\min} = 0$ KWh, and $B^{\max} = 100$ KWh. The buying prices of energy and carbon allowance $\alpha_{\text{tra}}^k, \alpha_{\text{res}}^k$ and α_{car}^k are generated from a uniform distribution in the range of $[0, 10]$, and the selling prices are $\beta_{\text{tra}}^k = 0.9\alpha_{\text{tra}}^k$, $\beta_{\text{res}}^k = 0.9\alpha_{\text{res}}^k$, and $\beta_{\text{car}}^k = 0.9\alpha_{\text{car}}^k$. The renewable energy arrivals e_s^k and task arrivals $R_{u,i}^t$ are also generated from a uniform distribution.

The following four benchmarks are compared. 1) DDPG_no_MA: Problem **P3** at the device layer is optimized by DDPG instead of MADDPG. 2) DDPG_single: All the optimization variables are determined by a single-layer DDPG. 3) TTMADRL_without_RES: This is TTMADRL with energy harvesting disabled. 4) Random: The optimization variables are randomly selected.

Fig. 3 presents a comparison of the average system costs among the considered algorithms. Clearly, the proposed TTMADRL converges to the lowest cost, which is 33.26%, 73.88%, 74.71% and 75.44% lower compared to

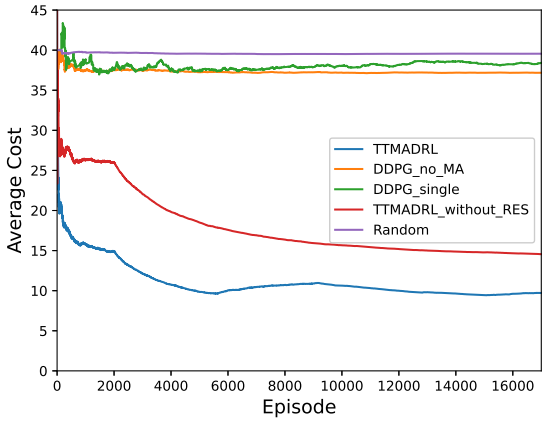

Fig. 3. Comparison of the time-average system costs.

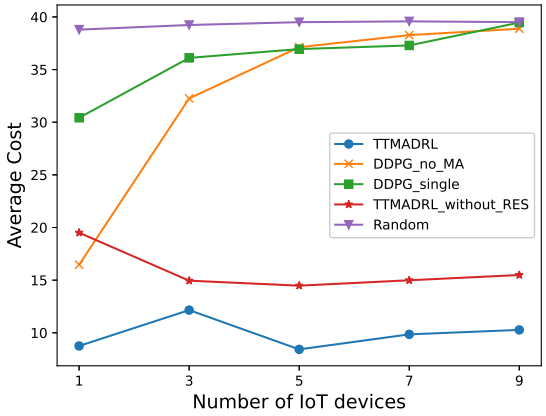

Fig. 4. Time-average system costs versus the number of devices.

TTMADRL_without_RES, DDPG_no_MA, DDPG_single, and Random, respectively. This is because the TTMADRL takes advantage of the dynamic energy/carbon prices and maximizes the utilization of RES by the two-layer multi-agent learning mechanism.

Fig. 4 illustrates the average costs under different numbers of devices. It is observed that as the number of devices grows, the system cost of TTMADRL changes marginally, exhibiting scalability and resilience to the increase of device number. In contrast, the costs of DDPG_no_MA and DDPG_single algorithms, which are not designed for multi-agent environments, increase rapidly with the device number.

Fig. 5 compares the task success rates of the considered algorithms under different numbers of devices. A task fails if the available energy or carbon allowance planned by the DRL agent is insufficient to support task execution. The proposed TTMADRL (with or without RES) can always maintain a 100% of task success rate as the number of devices grows, which demonstrates its robustness and reliability. The task success rates of DDPG_no_MA and DDPG_single, however, decline significantly when more devices share and compete for limited radio and computational resources.

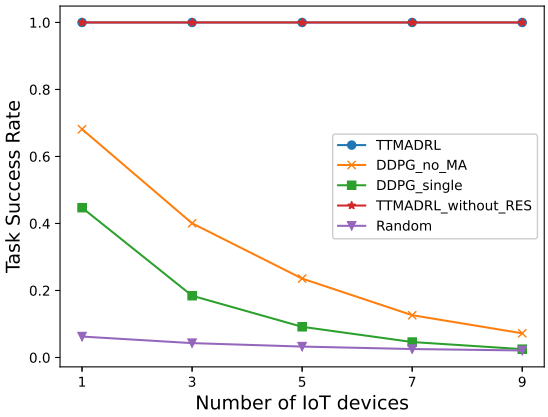

Fig. 5. Task success rate versus the number of devices.

VI. CONCLUSIONS

In this paper, we developed TTMADRL to minimize the system cost for MEC participating in integrated energy and carbon markets. The DDPG optimizes energy and carbon trading at the edge layer, while the MADDPG decides task offloading and CPU frequencies at the device layer with proved scalability. Simulations showed that the proposed TTMADRL can save the system cost by 75.44%, compared to its benchmarks.

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. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 2017.

[2] S. Hu, X. Chen, W. Ni, X. Wang, and E. Hossain, "Modeling and analysis of energy harvesting and smart grid-powered wireless communication networks: A contemporary survey," *IEEE Trans. Green Commun. Netw.*, vol. 4, no. 2, pp. 461–496, Jun. 2020.

[3] C. Feng, Y. Wang, Q. Chen, Y. Ding, G. Strbac, and C. Kang, "Smart grid encounters edge computing: Opportunities and applications," *Advances in Applied Energy*, vol. 1, p. 100006, Feb. 2021.

[4] D. Zhang, H. Zhu, H. Zhang, H. H. Goh, H. Liu, and T. Wu, "Multi-objective optimization for smart integrated energy system considering demand responses and dynamic prices," *IEEE Trans. Smart Grid*, vol. 13, no. 2, pp. 1100–1112, 2022.

[5] X. Chen, S. Chen, W. Ni, X. Wang, S. Zhang, S. Zhang, Y. Sun, S. Xu, and A. Jamalipour, "Optimal two-timescale configuration of mobile edge computing with mixed energy supply," *IEEE Trans. Smart Grid*, vol. 15, no. 5, pp. 4765–4778, Dec. 2024.

[6] X. Chen, H. Wen, W. Ni, S. Zhang, X. Wang, S. Xu, and Q. Pei, "Distributed online optimization of edge computing with mixed power supply of renewable energy and smart grid," *IEEE Trans. Commun.*, vol. 70, no. 1, pp. 389–403, Jan. 2022.

[7] H. Ma, Z. Zhou, X. Zhang, and X. Chen, "Toward carbon-neutral edge computing: Greening edge AI by harnessing spot and future carbon markets," *IEEE Internet Things J.*, vol. 10, no. 18, pp. 16 637–16 649, Sep. 2023.

[8] Z. Liu, J. Huang, R. Sun, and Y. Yu, "An optimal dispatch model for virtual power plant considering carbon trading and green certificate trading," in *Proc. ICPST*, 2023, pp. 703–708.

[9] Y. Yang, J. Shi, D. Wang, C. Wu, and Z. Han, "Identifying operation equilibrium in integrated electricity, natural gas, and carbon-emission markets," Oct. 2022, arXiv:2210.09813 [eess].

[10] X. Chen, Z. Chen, W. Ni, Z. Bai, and S. Zhang, "Joint user association and resource allocation for smart-grid-powered wireless networks under constrained carbon emission," *IEEE Wireless Commun. Lett.*, vol. 13, no. 11, pp. 3217–3221, 2024.