Modular Deep Reinforcement Learning for Multi-Workload Offloading in Edge Networks

Hongchang Ke[1] , **Yan Ding**[1,2] , **Lin Pan**[1] , **Yang Chen**[1] and **Jia Zhao** [1,2,3] *
[1]School of Computer Technology and Engineering, Changchun Institute of Technology of China
[2]College of Artificial Intelligence Technology, Changchun Institute of Technology of China
[3]School of Electronics Engineering and Computer Science, Peking University of China
{rj_khc, dingyan, panlin, chenyang, zhaojia}@ccit.edu.cn

Abstract

Dynamic edge networks revolutionize mobile edge computing by enabling real-time applications in intelligent transportation, augmented reality, and industrial Internet of Things (IoT). Efficient workload offloading in dynamic edge networks is crucial for addressing the increasing demands of time-varying workloads while contending with limited computational and communication resources. Existing deep reinforcement learning (DRL)-based offloading decision-making schemes are inadequate for managing scenarios involving multiple workloads and edge servers, particularly when faced with time-varying workload arrivals and fluctuating channel states. To this end, we propose a flexible module weighted fusion DRL framework (DRL-MWF) for scalable and robust multi-workload offloading in edge environments. Unlike traditional monolithic networks, DRL-MWF employs a weighted fusion modular architecture that adapts flexibly to diverse workload distributions. Specifically, DRL-MWF introduces a state representation and normalization strategy to model state and workload characteristics, enabling precise and adaptive decision-making. Furthermore, we design two key mechanisms: a weighted policy correction method to stabilize learning and a prioritized experience replay with weighted importance sampling to accelerate convergence by emphasizing critical transitions. Extensive evaluations on real-world datasets demonstrate that DRL-MWF consistently outperforms state-of-the-art baselines. These results reveal DRL-MWF's potential to transform workload offloading in next-generation edge computing systems, ensuring high performance in dynamic scenarios.

1 Introduction

In recent years, the proliferation of wireless devices (WDs) has led to an unprecedented increase in the generation of real-time workloads [Zhang and Debroy, 2023]. These WDs frequently encounter considerable challenges due to their limited computational power and battery life, rendering them incapable of processing all workloads locally [Chen *et al.*, 2024]. Mobile edge computing (MEC) has emerged as a promising solution, allowing for the offloading of workloads from WDs to nearby MEC servers [Yu *et al.*, 2024]. This policy improves computational efficiency, minimizes latency, and enhances the overall user experience [Qu *et al.*, 2024].

In dynamic MEC environment, the key problem lies in determining the offloading decision-making for workloads [Feng *et al.*, 2022] or resource allocation policy [Zhang *et al.*, 2023], which must take into account the prevailing computation and communication conditions of the MEC servers (such as channel state and energy availability) alongside time-varying workloads (like workload size and deadlines), with the objective of minimizing the cost of finishing workloads [Luo *et al.*, 2021]. The traditional heuristic [Almadhor *et al.*, 2022] or convex optimization algorithm [Tan *et al.*, 2022] can implement the workload offloading or resource allocation decision in the reliable MEC model environment. To this end, when faced with the complex environment of multiple workloads and multiple MEC servers, the aforementioned methods struggle to make near-optimal offloading or allocating decisions to minimize the system cost. Fortunately, deep reinforcement learning (DRL) emerges as an effective solution to these challenges, owing to its robust decision-making capabilities [Yang *et al.*, 2024].

Currently, there are many efforts dedicated to developing DRL-based computation offloading strategies, e.g., vehicular edge computing [Shi *et al.*, 2022; Hazarika *et al.*, 2022], collaborative edge computing [Ren *et al.*, 2024; Zhang *et al.*, 2024]. However, most related works focus on scenarios involving a single MEC server or address situations where the workloads exhibit low complexity. In addition, research has been conducted on workload offloading strategies involving multiple edge servers, as illustrated in studies [Xu *et al.*, 2023]. Nevertheless, these studies universally assume that there is no interaction among MEC servers and a static environment for MEC, making it challenging to address the demands posed by complex multi-workloads. Consequently, it is the general trend to study workload offloading in environments with numerous time-varying workloads and multiple edge servers, characterized by fluctuating channel states. Complex multi-workload offloading within a collaborative

*The corresponding author

framework of multiple MEC servers presents several significant challenges. Figure 1 illustrates the primary challenges. *The first is the heterogeneity of workloads.* Workloads generated by WDs exhibit diverse categories, sizes, required CPU cycles, deadlines, etc. This diversity poses significant challenges in tackling workloads as different characteristics demand varying computation resources and processing methods, making it difficult to manage and schedule them in a unified manner. *The second is the homogeneity of agents.* Neural network (NN) structures employed in DRL for agents tend to be similar, without considering the reuse of different networks for different workloads. This implies that agents may not perform optimally when confronted with diverse complex workloads, as they lack the flexibility to adequately address the specific requirements associated with different workload types. *The third is the inefficiency of action exploration.* In a dynamic MEC environment, random sampling or uniform sampling is not applicable and leads to high deviations. such inefficient sampling strategies may hinder the agents' ability to identify optimal action plans, as they do not accurately represent the true value of various actions within a complex environment, thereby impacting the system's overall performance and efficiency.

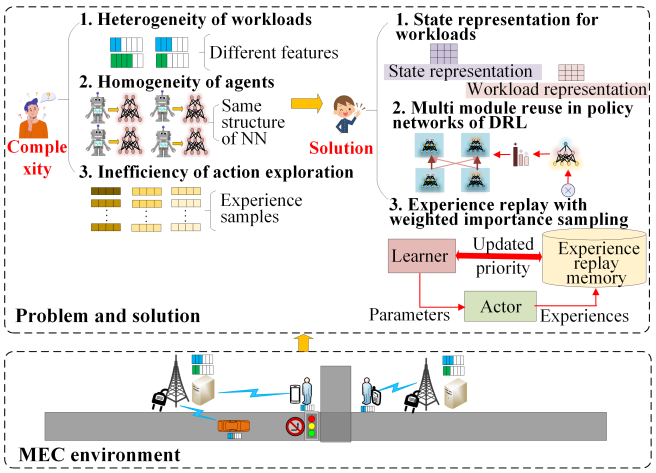


Figure 1: Issues of computation offloading in complex multi-workload, multi-MEC environment including: (1) The heterogeneity of workloads. (2) The homogeneity of agents. (3) The inefficiency of action exploration.

In terms of the above issues, we conceive and design the solution, summarized as follows: *State representation.* We perform state representation for the state features and workload features of the workloads, and delineate each workload individually. This approach allows for a more comprehensive and accurate depiction of workloads and their states, yielding valuable insights for subsequent decision-making and facilitating improved management of workload heterogeneity. *Multi-module reuse in policy networks.* We implement multi-module reuse in the policy networks of DRL. This enables the agent to flexibly select and combine appropriate modules according to the characteristics of different workloads, thereby enhancing the agent's ability to process different types of workloads and addressing the issues caused by the homogeneity of agents. *Prioritized experience replay with weighted importance sampling.* Agents can utilize experience samples more effectively during the learning process. Through weighted sampling of experiences and preferentially replaying those experiences that are more valuable for learning, it helps to improve the efficiency of action exploration, reduce sampling deviations, and enable the agent to quickly learn the optimal action strategy in complex environments.

To the best of our knowledge, we are the first to propose DRL-based state representation, policy network modularization and prioritized experience replay with importance sampling. A flexible module weighted fusion DRL with state representation for multi-workload offloading policy (DRL-MWF) in edge network is presented to learn the optimal offloading decision-making policy for minimizing the weighted average cost for finishing workloads. The main contributions of our work are summed up as follows:

- We propose a representation learning-based state representation method for multi-workload and multi-MEC environments. Different from the conventional image-based representation policy, we characterize and normalize the features of computation and communication resources for MEC server, as well as workload features of WDs respectively to adapt to the proposed DRL-MWF's policy network.
- We formulate DRL-MWF, and redesign its policy network on the basis of soft actior-critic (SAC) to execute offloading decision-making actions with adaptive module fusion. The learning efficiency of DRL-MWF can be enhanced through the reuse and composition of its modules.
- We develop a twin critic network that facilitates module composition to address the overestimation challenges associated with the policy network, arising from modular reuse and composition, thereby enhancing the stability of DRL-MWF. In addition, we propose a prioritized experience replay with weighted importance sampling for DRL-MWF, aimed at improving the tradeoff between exploration and exploitation, and adapting to multi-workload offloading decision-making contexts in diverse MEC server environments.
- We conduct comprehensive experiments to validate the convergence and advantage of DRL-MWF compared to five other benchmarks in terms of the average cumulative reward, the ratio of unfinished workloads.

2 Related Work

DRL focused on how the agent took a series of actions in an environment to maximize the expected cumulative reward by learning and approximating of intricate state representations as well as value functions from deep neural network [François-Lavet *et al.*, 2018]. DRL consisted of value-based scheme (deep Q-Network (DQN) [Mnih *et al.*, 2015], double deep Q-Network (DDQN) [Van Hasselt *et al.*, 2016], dueling deep Q-Network [Wang *et al.*, 2016]) and policy-based scheme (Actor-Critic [Peters and

Schaal, 2008], SAC [Haarnoja *et al.*, 2018], proximal policy optimization(PPO)[Schulman *et al.*, 2017]). DRL had been proved to be useful for controlling the discrete and continuous actions, such as: [Xiang *et al.*, 2023], [Yang *et al.*, 2020]. Because of its powerful decision-making ability, DRL was extensively applied in the field of MEC for workload offloading and resource allocation.

Currently, several studies had explored DRL-based workload offloading in the MEC environment. [Aghapour *et al.*, 2023] formulated a DRL-based policy to optimize computation offload and resource allocation decision-making selection. [Huang *et al.*, 2024] combined directed acyclic graphs (DAGs) and DRL to optimize the offloading scheme in the MEC framework. [Consul *et al.*, 2024] proposed federated reinforcement learning (FRL)-based offloading strategy to tackle the tasks from WDs. [Ling *et al.*, 2024] implemented PPO-based optimization method to address the computation resource allocation issue. However, the MEC environments set by the aforementioned works were relatively simple, and the multi time-varying workloads or channel states were not considered.

To sum up, we proposed DRL-MWF offloading scheme tailored for intricate multi-workload scenarios in the MEC environment. First, we considered the state representation for complex features of multi-workload and MEC environment. Different from most existing representation learning for complex image-based workloads, we could characterize the state of workloads and environments, thereby enabling DRL to accommodate multi-workload learning. Second, Multitasking was heterogeneous, but the WDs were similar. Assigning each WD or MEC server as an independent learning agent resulted in reduced learning efficiency. We designed network modularization and weighted combination for specific modules to improve the efficiency of DRL's learning. Third, we employed prioritized experience replay coupled with importance sampling to facilitate the exploration of actions taken by the scheduling agent, thereby enhancing the learning effect.

3 System Model and Problem Formulation

3.1 System Model

As illustrated in Fig.2, our focus is on the offloading of multiple workloads in an MEC environment that comprises various WDs and multiple MEC servers. There are W WDs and M MEC servers connected to M base stations (BSs) [1], meeting $\mathcal{W} = \{1, 2, \cdots, W\}$, $\mathcal{M} = \{1, 2, \cdots, M\}$. WD w generates different K sub-workloads $\mathcal{K}_w = \{1, 2, \cdots, K_w\}$, that must be processed within the stringent deadline. However, given the constraints of computing capacity and battery life, WDs cannot tackle all generated workload sets, a significant portion of the sub-workloads must be offloaded to the MEC servers for processing. MEC server m contains several core components: a computing unit,a communication unit and a workload scheduling unit. The computing unit is responsible for processing the received workloads in the buffer queue from WDs. The communication unit facilitates data transmission between the WDs and the MEC servers. The workload scheduling unit manages the scheduling of various workloads, taking into account the available computation resources of the current MEC server. Without loss of generality, WD w may select an appropriate offloading strategy to optimize system performance. A conventional method involves the complete offloading of all workloads to one MEC server; however, due to the intricate nature of these workload sets, we explore the option of utilizing edge collaborative processing. The sub-workloads in the workload set from each WD are assigned to several MEC servers, our attention shifts to their processing on multiple MEC servers, which encompasses resource distribution of MEC servers for finishing workloads. To this end, $x_{k_w,m}$ is introduced to denote the offloading decision variable, and $y_{k_w,m}$ signifies the ratio of computational resources allocated for the execution of sub-workload k_w generated by WD w. The decision variable $x_{k_w,m} \in \{0, 1, 2, \cdots, M\}$ means whether the workload generated by WD m can be executed locally (when $x_{k_w,m} = 0$) or offloaded to MEC server m (when $x_{k_w,m} = m$). The decision variable $y_{k_w,m} \in [0, 1]$ is the ratio of allocated computational resources at MEC server m.

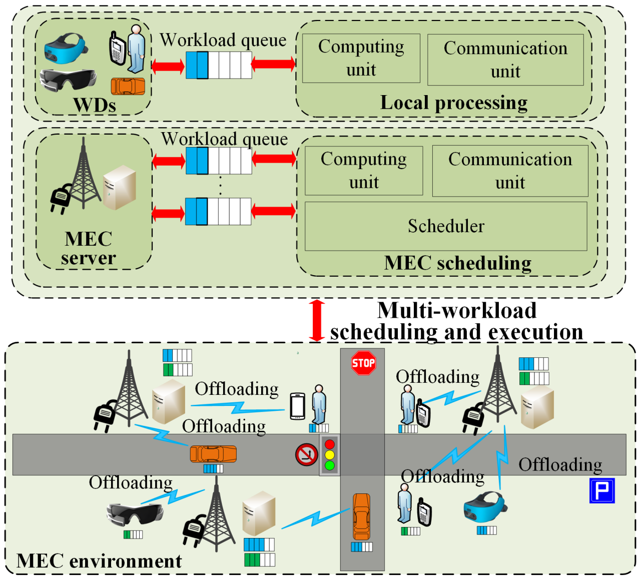


Figure 2: Multi-workload offloading with multi-WD and multi-edge server in MEC environment.

Communication Model

Due to the limitation of the computation capacity of the local WD, workloads can be transmitted (offloaded) to the MEC server via the wireless network for execution. Initially, we discretize a continuous time interval T into discrete time slots t, meeting $t \in T$. Similar to [Wu *et al.*, 2019], we consider that each MEC server operates on one of the bandwidth subcarriers that are accessible. As a result, the communication method is established as orthogonal frequency division multiple access (OFDMA). The total bandwidth available between the WDs and MEC server m is designated as B_m. According to Shannon's theorem, the transmission rate between WD w

[1]Note that each MEC server is associated with a BS; in this context, we refer to them as MEC servers instead of BSs.

and MEC server m in time slot t can be given by

$$R_{w,m}(t) = B_m \log_2 \left(1 + \frac{p_{w,m}(t) h_{w,m}(t)}{\sigma_m(t)^2}\right), \quad (1)$$

where $p_{w,m}(t)$ represents the transmission power for offloading workloads. $h_{w,m}$ stands for the channel states between WD w and MEC server m, meeting the Gaussian Markov autoregressive model [Ke *et al.*, 2020]:

$$h_{w,m}(t+1) = \rho_m h_{w,m}(t) + \sqrt{1-\rho_m^2} e_m(t), \quad (2)$$

where ρ_m means the path-loss coefficient, describing signal attenuation as it propagates between WD w and MEC m, which is close to 1. $e_m(t)$ is an error term that follows a complex Gaussian distribution, independent of the channel state.

Computation Model

We redefine the sub-workload generated by WD w in the time slot t as the tuple $k_w(t) = \langle k_w^s(t), k_w^c(t), k_w^D(t)\rangle$, where $k_w(t) \in \mathcal{K}_w$. $k_w^s(t)$ denotes the size of workload $k_w(t)$, $k_w^s(t)$ represents the number of required CPU cycles for $k_w(t)$, and $k_w^D(t)$ means the deadline for finishing $k_w(t)$.

Local Computation. As described in section 3.1, while $x_{w,m} = 0$, the processor of WD w can accomplish the sub-workload $k_w(t)$ it generates. To this end, the latency $l_{k_w}^{lc}(t)$ consists of the local execution latency $l_{k_w}^{lc,exe}(t)$ and the queue waiting latency $l_{k_w}^{lc,queue}(t)$, which can be calculated by

$$l_{k_w}^{lc,exe}(t) = \frac{k_w^s(t) k_w^c(t)}{F^w}, l_{k_w}^{lc,queue}(t) = \frac{Q_w(t) k_w^c(t)}{F^w}, \quad (3)$$

where $Q_{k_w}(t)$ is the size of workloads queue at WD w in the time slot t, and F^w is the available computation frequency of WD w.

The energy consumption $e_{k_w}^{lc}(t)$ can be given by

$$e_{k_w}^{lc,exe}(t) = \kappa_w (F^w)^2 k_w^s(t) k_w^c(t), \quad (4)$$

where κ_w is the effective switching capacitance who depends on the computation ability of local processor. Since the queue waiting latency are far less the execution latency, which can be ignored [Van Huynh *et al.*, 2022].

Offloading to MEC server. While $x_{k_w,m} \neq 0$, it is necessary to offload the sub-workload $k_w(t)$ from WD w to MEC server for execution. Therefore, the latency $l_{k_w,m}^{off}(t)$ consists of the transmission latency $l_{k_w,m}^{off,tran}(t)$, the execution latency $l_{k_w,m}^{off,exe}(t)$ and the queue waiting latency $l_{k_w,m}^{off,queue}(t)$, which can be calculated by

$$\begin{aligned} l_{k_w,m}^{off,tran}(t) &= \frac{k_w^s(t)}{R_{w,m}(t)}, l_{k_w,m}^{off,exe}(t) = \frac{k_w^s(t) k_w^c(t)}{y_{k_w,m} F^m}, \\ l_{k_w,m}^{off,queue}(t) &= \frac{Q_{k_w,m}^s(t) k_w^c(t)}{y_{k_w,m} F^m}. \end{aligned} \quad (5)$$

To this end, the energy consumption $e_m^{off}(t)$ includes the transmission energy consumption $e_m^{off,tran}(t)$ and the execution energy consumption $e_{k_w,m}^{off,exe}(t)$ can be derived by

$$\begin{aligned} e_{k_w,m}^{off,tran}(t) &= p_{w,m}(t) l_{k_w,m}^{off,tran}(t), \\ e_{k_w,m}^{off,exe}(t) &= \kappa_m (y_{k_w,m} F^m)^2 k_w^s(t) k_w^c(t). \end{aligned} \quad (6)$$

3.2 Problem Formulation

In this study, we define the optimization objective as finding the optimal workload offload selection $x_{k_w,m}$ and the ratio of allocated computation resource $y_{k_w,m}$ for the MEC server throughout the whole time period T, with the aim of minimizing the weighted average latency and energy consumption costs of the MEC system. Hence, the optimization objective function can be derived as

$$C(t) = \beta_1 l(t) + \beta_2 e(t), \quad (7)$$

where $l(t)$ means the average latency of all W WDs, meeting $l(t) = [\sum_{w \in W} \sum_{k_w \in K_w} (1 - x_{k_w,m}) l_{k_w}^{lc}(t) + (x_{k_w,m}/m) l_{k_w,m}^{off}(t)]/W$, and $e(t)$ is the average energy consumption. β_1, β_2 are the weighted parameters for latency cost and energy consumption cost, meeting $\beta_1 + \beta_2 = 1$.

Furthermore, the optimization problem can be described as

$$\mathbf{P1}: \quad \min_{(x_{k_w,m}, y_{k_w,m})} \mathbb{E}\left[\lim_{t\to\infty} \frac{1}{T} \sum_{t=1}^{T} C(t)\right], \quad (8)$$

$$\text{s.t.} \quad x_{k_w,m} \in [0, 1, \cdots, M], \forall k_w \in K_w, m \in M, \quad \text{(C1)}$$

$$0 < y_{k_w,m} \leq 1, \forall k_w \in K_w, m \in M, \quad \text{(C2)}$$

$$F^w \leq F^{w,max}, \forall k_w \in K_w, \quad \text{(C3)}$$

$$l_{k_w,m}^{off}(t) \leq K_w^D(t), \forall k_w \in K_w. \quad \text{(C4)}$$

4 Multi-workload Offloading Policy with DRL-MWF

As outlined in Section 3.2, **P1** is an NP-hard problem, posing significant challenges for conventional convex optimization techniques or heuristic methods. The complexity arises from the dynamic nature of the MEC environment, characterized by factors such as the time-varying arrival workloads from WDs and varying channel states between the WDs and MEC servers. Consequently, we transform the optimization problem **P1** into a Markov decision process (MDP) and define the tuple $\langle S, A, P, R, \gamma\rangle$. The state $s(t)$ in state space S in the time slot t can be characterized as

$$\begin{aligned} s(t) = \{& k_w^s(t), k_w^c(t), k_w^D(t), B_m, F^w, F^m, \\ & Q_{k_w}(t), Q_{k_w,m}(t) \,|k_w \in K_w, w \in W\}. \end{aligned} \quad (9)$$

Hence, The action $a(t)$ in action space A in the time slot t can be defined as

$$a(t) = \{x_{k_w,m}, y_{k_w,m} | k_w \in K_w, w \in W\}. \quad (10)$$

Furthermore, the reward $r(t) \in R$ can be defined as $r(t) = -C(t) + \mathbb{I}(.)\mathbf{p}$, where $\mathbb{I}(.)$ is the indicator function which means whether the workload is completed within the deadline. The term $\mathbf{p}$ represents the penalty factor associated with any incomplete workload. In this way, the problem of minimizing the weighted average cost is transformed into obtaining the maximum expected cumulative reward; **P1** can be transformed into **P2**:

$$\mathbf{P2}: \quad \arg\max_{\pi} \mathbb{E}_{a\sim\pi}\left[\lim_{t\to\infty} \frac{1}{T} \sum_{t=1}^{T} \gamma^{t-1} r(t)\right]. \quad (11)$$

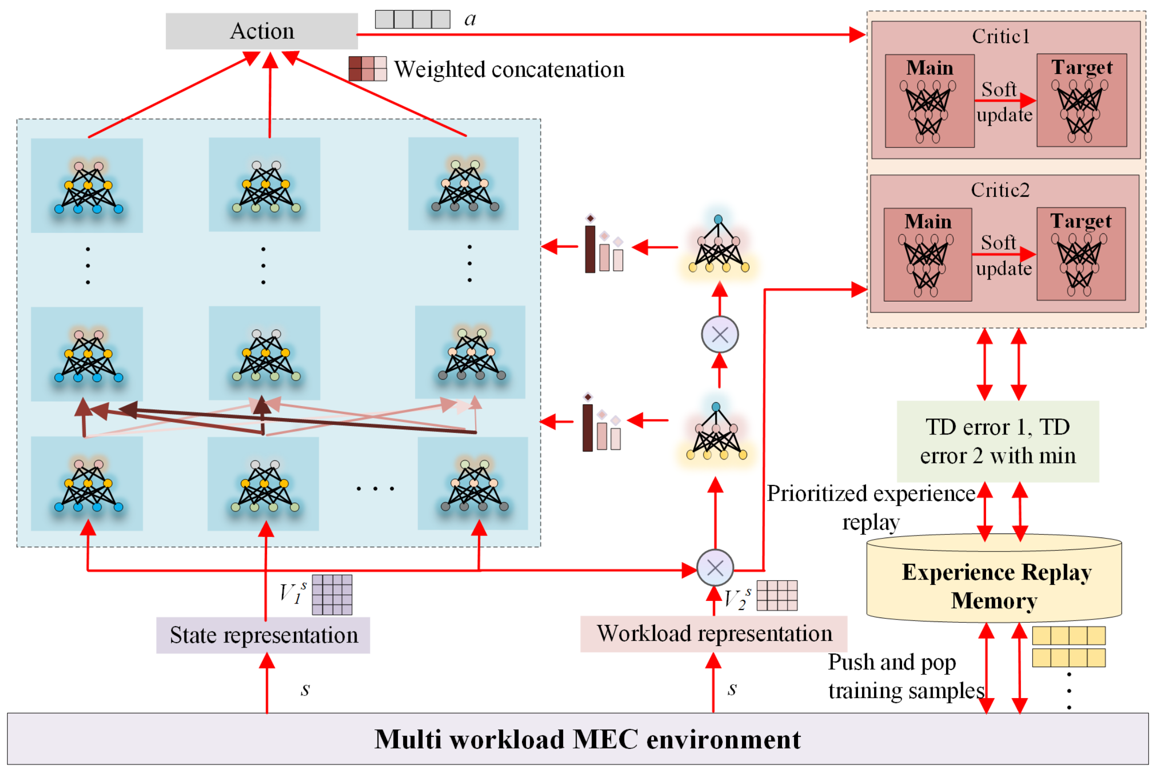


Figure 3: Structure of the proposed DRL-MWF. On the left is the state representation and module fusion of the policy network, and on the right is the twin critic networks and the priority experience replay with importance sampling.

Due to the change of environment, we cannot obtain the state transition probability matrix P. Therefore, we convert the MDP to DRL, and propose the DRL-MWF according to the multi-workload inputs and dynamic channel states.

Figure 3 illustrates the DRL-MWF framework, highlighting state representation and module fusion strategies. Building upon SAC, we initially modularize the actor network for multi-task complex MEC scenarios, and design a twin critic network to solve the overestimation issue while enhancing the stability of the evaluation strategy for the multi-module of actor. In terms of policy network, there are two networks: a state representation network and a workload schedule network. In each time slot, the state $s(t)$ is divided into two parts, the state $s_1(t)$ related to the environment and the state $s_2(t)$ related to the workload. We represent $s_1(t)$ and $s_2(t)$ as V_1^s and V_2^s with two-layer multilayer perceptron (MLP) and one-layer MLP, one fully connected layer, respectively. For the represented state vector V_1^s, we implement AvgL1Norm to prevent representation collapse and unify the vector scale by $\frac{V_1^s}{||V_1^s||^1/D}$, where D is the dimension. To this end, the schedule network obtains the represented and normalized states V_1^s and V_2^s with D-dimension vector and further output the probability vector by ReLU function. The probability vectors for the first layer and the $l+1$-layer can be computed as $p^1 = \mathrm{FC}_d^1(\mathrm{ReLU}(V_1^s, V_2^s))$ and $p^{l+1} = \mathrm{FC}_d^l(\mathrm{ReLU}(p^l V_1^s, V_2^s))$, respectively. In terms of policy network, there are L layers and K modules in each layer and the state representation for module k in layer l can be written as $V_j^l = \sum_{k\in K} \mathrm{S}(p^{l-1}(j,k))\mathrm{ReLU}(W_k^{l-1}V_k^{l-1})$, where S is the Softmax function and W_k^{l-1} is the weighted matrix. This means that we can select modules of each layer based on probability, realize module reuse, and improve the efficiency of network training. Finally, the last layer outputs the action by mean and variance.

To ensure the reuse efficiency of the proposed DRL-MWF's policy module, we propose the following two improvement strategies to enhance the network training effectiveness and algorithm performance.

First, leveraging the twin critic networks of SAC, we modify the maximum entropy objective function and incorporate stability constraints to minimize the subsequent loss function aimed at preventing overestimation:

$$\mathcal{L}(Q_i) = \mathbb{E}_{(s(t),a(t),r(t),s'(t))}\left[(y - Q_i(s(t),a(t)))^2\right], \quad (12)$$

where $y = r(t) + \gamma \cdot \min_{i=1,2} Q_j(s'(t), \mu(s'(t)) + \mathcal{N})$.

Therefore, the objective function of policy network is updated by $J(\pi) = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi}[Q(s,a) - \alpha \log \pi(a|s) - \lambda_d \Delta(s,a)]$, where λ_d means the weighted adjusting coefficient. $\Delta(s,a)$ is the deviation correction term which can be estimated by multiple sampling, satisfying $\Delta(s,a) = \frac{1}{N}\sum_{i\in N}(\max(Q_{1i}^+, Q_{2i}^+) - Q(s_t,a_t))$. Q_{1i}^+, Q_{2i}^+ stand for the estimations of the Q function. This approach allows for the mitigation of Q-function overestimation, thereby enhancing the stability of policy optimization.

Second, beyond regulating the balance between exploration and exploitation in the policy through the temperature parameter α of the traditional SAC, we also develop a priority experience replay mechanism utilizing weighted importance sampling to further enhance policy exploration by:

$$\mathcal{L}(Q) = \sum_{d\in\mathcal{D}} p_d \cdot \mathcal{L}_d, \tag{13}$$

where $\mathcal{L}_d$ means the loss of sample d. p_d signifies the priority of sample which can be obtained through TD Error as $p_d = |\Delta_d| + \epsilon$. Δ_d stands for the TD error of sample d, ϵ is a small constant used to ensure that the priority is not zero.

We employ the sampled experiences to update the learning process; however, it is necessary to adjust the learning rate. Consequently, we introduce weighted importance sampling to maintain the unbiased nature of the learning process. the actual importance weight associated with the sampled experience is denoted as ω_d, which can be expressed as,

$$\omega_d = (\frac{1}{N} \cdot \frac{1}{p_d})^{\beta}, \tag{14}$$

where β is another hyperparameter used to control the scaling of the importance weight. During the actual training process, the loss function is adjusted by incorporating the weighted TD error and loss, updated by $\mathcal{L}(Q) = \sum_{d\in\mathcal{D}} \omega_d \cdot \mathcal{L}_d$.

5 Experiments

In this section, we first introduce the datasets for multi-workload, multi-edge server in dynamic MEC environment, then describe the experiment setups, metrics and benchmarks. Finally, we analysis evaluation results from the convergence and performance of DRL-MWF.

5.1 Datasets

We utilize a universal EUA dataset [Lai *et al.*, 2018] comprising 125 MEC Servers/BSs and 816 WDs located in Melbourne street district. We initialize 5 BSs equipped with MEC servers and 50 WDs from EUA dataset. For the size $k_w^s(t)$ and required CPU cycles $k_w^c(t)$ of workloads, we employ the UE application data [Rojas, 2023] and the SPEC CPU 95 benchmark [Phansalkar *et al.*, 2005] to simulate multiple workloads in the MEC environment.

5.2 Environment Settings and Metrics

Regarding the workload distribution, the sizes of three sub-workloads are set as $1 \sim 4$Mb, $3 \sim 6$Mb, $5 \sim 8$Mb from UE application dataset. Additionally, the required CPU cycles per bit for three sub-workloads are set as 800, 1000, 1200 from SPEC CPU 95 benchmark dataset. Table 1 lists additional key parameters in the environment. For DRL-MWF, the parameters include the number of training episodes, the learning rates for both the actor and critic, the discount factor, the size of experience memory, the number of modules, and the number of layers in the actor's network, which are set to 1000, 0.0001, 0.001, 0.9, 10000, 3, 2, respectively.

To assess the convergence of DRL-MWF, we utilize the average cumulative reward [Ke *et al.*, 2022], represented as $R^{\text{ACR}} = \frac{1}{T}\sum_{t\in T} r(t)$. In terms of the performance comparison of DRL-MWF against benchmarks, we evaluate not only R^{ACR}, but also the average latency cost $C^{\text{ALC}} = \frac{1}{T}\sum_{t\in T} l(t)$, the average energy consumption cost $C^{\text{AEC}} = \frac{1}{T}\sum_{t\in T} e(t)$, and the ratio of unfinished workloads U^{RUW} calculated by dividing the number of unfinished workloads by the total workloads [Tang and Wong, 2020].

Parameter	**description**	**Value**
K_w	Number of sub-workloads	3
B_m	Bandwidth	10MHz
$p_{w,m}$	Transmission power	1W
T	Number of time slots	100
ρ_m	path-loss	0.95
$\sigma_m(t)$	AWGN	-100dBm
Q_w, Q_m	Size of queue at WD/MEC	50/500
$\mathbf{p}$	Penalty factor	$k_w^D(t)$

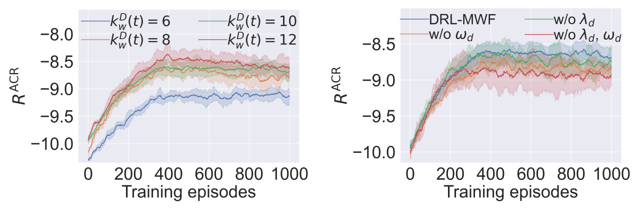
Table 1: List of the main parameters

5.3 Benchmarks

We introduce five benchmarks for comparative analysis: (1) DQN-LSTM [Tang and Wong, 2020], where Dueling DQN is employed for MEC selection decisions, and LSTM is used to forecast the MEC server's task queue information. (2) MCO-DRL [Ke *et al.*, 2022], which combines DDQN and gated recurrent unit (GRU) to facilitate workload offloading to the MEC server while estimating both the workload size queue. (3) SAC [Shi *et al.*, 2020], which proposes a computation offloading policy based on SAC aimed at reducing the average latency of tasks. (4) PPO [Shang *et al.*, 2024], which implements a computation offloading policy utilizing PPO to minimize the weighted costs. (5) TD7 [Fujimoto *et al.*, 2023], an emerging DRL strategy designed for multi-task management, is applied for the first time in MEC environment.

5.4 Evaluation Results

Convergence analysis of DRL-MWF. Firstly, we analyze the convergence of DRL-MWF. R^{ACR} represents the average cumulative reward with penalty terms $\mathbf{p}$ for unfinished workloads, enabling the evaluation of DRL-MWF's convergence.



(a) Deadline $k_w^D(t)$ (b) Weighted factors λ_d, ω_d

Figure 4: Convergence of DRL-MWF. (a) Deadline $k_w^D(t)$ for finishing workload $k_w(t)$ generated by WD w. (b) Proposed two key mechanisms: the weighted policy correction factor λ_d and importance sampling factor ω_d for prioritized experience replay.

Figure 4 reveals the convergence of DRL-MWF under different parameters. As shown in Fig.4(a), DRL-MWF can

Metrics	Average cumulative reward R^{ACR}±95% confidence interval (CI)						
Datasets(EUA)	**MEC2-WD10**	**MEC3-WD15**	**MEC3-WD30**	**MEC5-WD25**	**MEC5-WD50**	**MEC8-WD40**	**MEC8-WD80**
DQN-LSTM	-10.62±0.012	-10.70±0.012	-10.72±0.015	-10.66±0.014	-10.96±0.015	-11.23±0.015	-13.34±0.016
MCO-DRL	-9.91±0.010	-9.87±0.013	-9.92±0.016	-9.97±0.015	-10.01±0.012	-10.34±0.011	-12.42±0.016
SAC	-9.37±0.010	-9.63±0.011	-9.88±0.013	-10.25±0.012	-10.42±0.012	-10.37±0.014	-13.76±0.017
PPO	-10.02±0.013	-9.95±0.015	-10.26±0.013	-10.96±0.015	-10.31±0.014	-10.34±0.012	-12.82±0.018
TD7	-9.36±0.015	-9.41±0.012	-9.53±0.015	-9.39±0.012	-9.60±0.013	-9.58±0.011	-11.13±0.013
DRL-MWF	-8.58±0.013	-8.24±0.014	-8.70±0.011	-8.39±0.009	-8.88±0.011	-8.77±0.009	-9.72±0.012

Table 2: The performance of different comparison algorithms in different MEC environment using EUA datasets. Each algorithm undergoes 10 epoches of training under 7 different numbers of MEC servers and WDs, with 10 random seeds and 1000 episodes in one epoch. The average cumulative reward R^{ACR}±95% CI as the performance evaluation.

converge under four different deadline settings which reflects the effectiveness of the proposed modular DRL. When $k_w^D(t) = 6$, a significant number of workloads fail to be completed within the stringent deadline, leading to latency penalties and a decrease in R^{ACR}. As $k_w^D(t)$ gradually increases from 8 to 12, MEC servers can fully schedule their own resources, which in turn enhances the workload completion ratio and leads to an increase in R^{ACR}. $k_w^D(t)$ is set as 10 in this work. As revealed in Fig.4(b), we compare DRL-MWF with the weighted policy correction factor λ_d and importance sampling factor ω_d for prioritized experience replay and without λ_d and ω_d. Although DRL-MWF can converge without λ_d and ω_d due to the modular structure, the performance of DRL-MWF without λ_d and ω_d is worse than that of DRL-MWF with λ_d and ω_d in terms of R^{ACR}, which verifies the key of introducing two important mechanisms[2].

Performance comparison with benchmarks. Table 2 illustrates the performance of benchmarks and our proposed DRL-MWF in terms of average cumulative rewards $R^{\text{ACR}} \pm 95\%$ confidence interval (CI). The performance of each algorithm varies across different combinations of MEC servers and WDs. For instance, in dataset with MEC2-WD10, the proposed DRL-MWF performs best with R^{ACR} of -8.58±0.013 because of the advantages of state representation and module grouping. As the configuration of datasets changes, the performance rankings of the algorithms also change. However, the performance of DRL-MWF with weighted modules is relatively stable across various configurations, and its R^{ACR} remains at a relatively high level compared to other benchmarks. Priority experience replay with importance sampling and maximum entropy revision mechanism ensure the effectiveness and stability of DRL-MWF.

Figure 5 illustrates the comparison on six algorithms with different probability of arriving workloads. From Fig.4(a), DRL-MWF has a greater R^{ACR} in any cases. With an increase in the probability of incoming workloads, R^{ACR} of DRL-MWF does not show a sharply decreasing trend due to the state representation and module fusion. The relationship between C^{ALC} and the probability of arriving workloads is depicted in Fig.5(b). Compared with other benchmarks, DRL-MWF shows relatively high R^{ACR} at increased probability of arriving workloads; however, this advantage diminishes progressively as the probability declines. Fig.5(c) presents C^{AEC} of six algorithms. DRL-MWF retains lower C^{AEC} throughout the process, indicating its advantage in energy conservation. Fig.5(d) illustrates that the probability of arriving workloads influences U^{RUW}. DRL-MWF shows superior performance in controlling the ratio of unfinished workloads under different probability of arriving workloads.

[2]DRL-MWF focuses on exploration at the beginning of training, setting λ_d to 0.2. The over-estimation of Q-value is noticeable. λ_d gradually increases from 0.2 to 0.7. For the importance factor ω_d, β in Eq.14 is initially set as 0.5 to enhance exploration, then gradually increased to prioritize experiences, finally reaching 1.

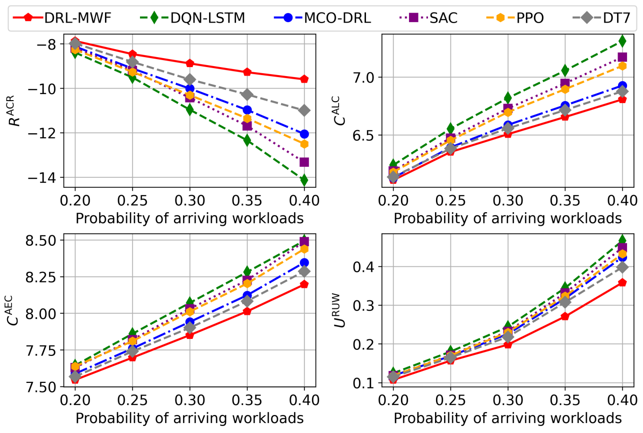


Figure 5: Comparison on six algorithms with different probability of arriving workloads. (a) Average cumulative reward R^{ACR}. (b) Average latency cost C^{ALC}. (c) Average energy cost C^{AEC}. (d) Ratio of unfinished workloads U^{RUW}.

6 Conclusion

In this work, to address the state representation of heterogeneous workloads, multi-agent training difficulties, and efficiency of action exploration of DRL-based offloading methods in multi-workload MEC environment, we proposed a flexible modular DRL with state representation for multi-workload offloading policy (DRL-MWF). DRL-MWF can implement state representation and normalization, and modularize the policy network, then utilize policy function weighted correct and priority experience replay with importance sampling to improve the efficiency and stability of training. Experiment results show that DRL-MWF can converge in a certain number of episodes and whose performance is better than that of DRL-based workload offloading schemes.

Acknowledgments

This work was supported by the Outstanding Young Science and Technology Talents Project of Jilin Province Science and Technology Development Plan(20240602004RC), the Natural Science Foundation of Jilin Province Science and Technology Development Plan (20250102246JC), the Key R & D Project of Jilin Province Science and Technology Development Plan (20240302080GX) and the Key Scientific Research Project of Jilin Provincial Department of Education (JJKH20251146KJ).

References

[Aghapour *et al.*, 2023] Zahra Aghapour, Saeed Sharifian, and Hassan Taheri. Task offloading and resource allocation algorithm based on deep reinforcement learning for distributed ai execution tasks in iot edge computing environments. *Computer Networks*, 223:109577, march 2023.

[Almadhor *et al.*, 2022] Ahmad Almadhor, Abdullah Alharbi, Ahmad M Alshamrani, Wael Alosaimi, and Hashem Alyami. A new offloading method in the green mobile cloud computing based on a hybrid meta-heuristic algorithm. *Sustainable Computing: Informatics and Systems*, 36:100812, December 2022.

[Chen *et al.*, 2024] Hualong Chen, Yuanqiao Wen, Yaming Huang, Changshi Xiao, and Zhongyi Sui. Edge computing enabling internet of ships: A survey on architectures, emerging applications, and challenges. *IEEE Internet of Things Journal*, November 2024.

[Consul *et al.*, 2024] Prakhar Consul, Ishan Budhiraja, Deepak Garg, Neeraj Kumar, Ramendra Singh, and Ahmad S Almogren. A hybrid task offloading and resource allocation approach for digital twin-empowered uav-assisted mec network using federated reinforcement learning for future wireless network. *IEEE Transactions on Consumer Electronics*, 70(1):3120–3130, January 2024.

[Feng *et al.*, 2022] Chuan Feng, Pengchao Han, Xu Zhang, Bowen Yang, Yejun Liu, and Lei Guo. Computation offloading in mobile edge computing networks: A survey. *Journal of Network and Computer Applications*, 202:103366, June 2022.

[François-Lavet *et al.*, 2018] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, Joelle Pineau, et al. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, November 2018.

[Fujimoto *et al.*, 2023] Scott Fujimoto, Wei-Di Chang, Edward J Smith, Shixiang Shane Gu, Doina Precup, and David Meger. For sale: state-action representation learning for deep reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 61573–61624, New Orleans LA USA, December 2023. ACM.

[Haarnoja *et al.*, 2018] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870, Stockholm, Sweden, July 2018. PMLR.

[Hazarika *et al.*, 2022] Bishmita Hazarika, Keshav Singh, Sudip Biswas, and Chih-Peng Li. Drl-based resource allocation for computation offloading in iov networks. *IEEE Transactions on Industrial Informatics*, 18(11):8027–8038, April 2022.

[Huang *et al.*, 2024] Chong Huang, Gaojie Chen, Pei Xiao, Yue Xiao, Zhu Han, and Jonathon A Chambers. Joint offloading and resource allocation for hybrid cloud and edge computing in sagins: A decision assisted hybrid action space deep reinforcement learning approach. *IEEE Journal on Selected Areas in Communications*, 42(5):1029–1043, May 2024.

[Ke *et al.*, 2020] Hongchang Ke, Jian Wang, Lingyue Deng, Yuming Ge, and Hui Wang. Deep reinforcement learning-based adaptive computation offloading for mec in heterogeneous vehicular networks. *IEEE Transactions on Vehicular Technology*, 69(7):7916–7929, May 2020.

[Ke *et al.*, 2022] Hongchang Ke, Hui Wang, Weijia Sun, and Hongbin Sun. Adaptive computation offloading policy for multi-access edge computing in heterogeneous wireless networks. *IEEE Transactions on Network and Service Management*, 19(1):289–305, March 2022.

[Lai *et al.*, 2018] Phu Lai, Qiang He, Mohamed Abdelrazek, Feifei Chen, John Hosking, John Grundy, and Yun Yang. Optimal edge user allocation in edge computing with variable sized vector bin packing. In *Service-Oriented Computing: 16th International Conference*, pages 230–245, Cham, November 2018. Springer.

[Ling *et al.*, 2024] Chengfang Ling, Kai Peng, Shangguang Wang, Xiaolong Xu, and Victor CM Leung. A multi-agent drl-based computation offloading and resource allocation method with attention mechanism in mec-enabled iiot. *IEEE Transactions on Services Computing*, 17(6):3037–3051, November 2024.

[Luo *et al.*, 2021] Quyuan Luo, Changle Li, Tom H Luan, and Weisong Shi. Minimizing the delay and cost of computation offloading for vehicular edge computing. *IEEE Transactions on Services Computing*, 15(5):2897–2909, March 2021.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, February 2015.

[Peters and Schaal, 2008] Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, March 2008.

[Phansalkar *et al.*, 2005] Aashish Phansalkar, Ajay Joshi, Lieven Eeckhout, and Lizy Kurian John. Measuring program similarity: Experiments with spec cpu benchmark suites. In *IEEE International Symposium on Performance Analysis of Systems and Software*, pages 10–20, Austin, TX, USA, March 2005. IEEE.

[Qu *et al.*, 2024] Long Qu, An Huang, Junqi Pan, Cheng Dai, Sahil Garg, and Mohammad Mehedi Hassan. Deep reinforcement learning-based multireconfigurable intelligent surface for mec offloading. *International Journal of Intelligent Systems*, 2024(1):2960447, July 2024.

[Ren *et al.*, 2024] Tao Ren, Zheyuan Hu, Jianwei Niu, Weikun Feng, and Hang He. M3off: Module-compositional model-free computation offloading in multi-environment mec. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*, pages 2249–2258, Vancouver, BC, Canada, May 2024. IEEE.

[Rojas, 2023] Juan Sebastián Rojas. Ip network traffic flows labeled with 75 apps. https://www.kaggle.com/datasets/jsrojas/ip-network-traffic-flows-labeled-with-87-apps, 2023. Accessed: 2023-04-22.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Shang *et al.*, 2024] Ce Shang, Youliang Huang, Yan Sun, and Mohsen Guizani. Joint computation offloading and service caching in mobile edge-cloud computing via deep reinforcement learning. *IEEE Internet of Things Journal*, 11(24):40331–40344, August 2024.

[Shi *et al.*, 2020] Jinming Shi, Jun Du, Jingjing Wang, Jian Wang, and Jian Yuan. Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 69(12):16067–16081, December 2020.

[Shi *et al.*, 2022] Jinming Shi, Jun Du, Yuan Shen, Jian Wang, Jian Yuan, and Zhu Han. Drl-based v2v computation offloading for blockchain-enabled vehicular networks. *IEEE Transactions on Mobile Computing*, 22(7):3882–3897, July 2022.

[Tan *et al.*, 2022] Kaige Tan, Lei Feng, György Dán, and Martin Törngren. Decentralized convex optimization for joint task offloading and resource allocation of vehicular edge computing systems. *IEEE Transactions on Vehicular Technology*, 71(12):13226–13241, December 2022.

[Tang and Wong, 2020] Ming Tang and Vincent WS Wong. Deep reinforcement learning for task offloading in mobile edge computing systems. *IEEE Transactions on Mobile Computing*, 21(6):1985–1997, November 2020.

[Van Hasselt *et al.*, 2016] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, JPhoenix, Arizona USA, February 2016. AAAI.

[Van Huynh *et al.*, 2022] Dang Van Huynh, Van-Dinh Nguyen, Saeed R Khosravirad, Vishal Sharma, Octavia A Dobre, Hyundong Shin, and Trung Q Duong. Urllc edge networks with joint optimal user association, task offloading and resource allocation: A digital twin approach. *IEEE Transactions on Communications*, 70(11):7669–7682, September 2022.

[Wang *et al.*, 2016] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003, New York, NY, USA, July 2016. PMLR.

[Wu *et al.*, 2019] Yuhang Wu, Yuhao Wang, Fuhui Zhou, and Rose Qingyang Hu. Computation efficiency maximization in ofdma-based mobile edge computing networks. *IEEE Communications Letters*, 24(1):159–163, October 2019.

[Xiang *et al.*, 2023] Yanfei Xiang, Xin Wang, Shu Hu, Bin Zhu, Xiaomeng Huang, Xi Wu, and Siwei Lyu. Rmbench: Benchmarking deep reinforcement learning for robotic manipulator control. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1207–1214, Detroit, MI, USA, October 2023. IEEE.

[Xu *et al.*, 2023] Changfu Xu, Yupeng Li, Xiaowen Chu, Haodong Zou, Weijia Jia, and Tian Wang. Smcoedge: Simultaneous multi-server offloading for collaborative mobile edge computing. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 73–91, Tianjin, China, October 2023. ACM.

[Yang *et al.*, 2020] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. *Advances in Neural Information Processing Systems*, 33:4767–4777, December 2020.

[Yang *et al.*, 2024] Ning Yang, Shuo Chen, Haijun Zhang, and Randall Berry. Beyond the edge: An advanced exploration of reinforcement learning for mobile edge computing, its applications, and future research trajectories. *IEEE Communications Surveys & Tutorials*, early access, 2024.

[Yu *et al.*, 2024] Di Yu, Xin Du, Linshan Jiang, Wentao Tong, and Shuiguang Deng. Ec-snn: Splitting deep spiking neural networks for edge devices. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 5389–5397, Jeju, South Korea, August 2024. Morgan Kaufmann.

[Zhang and Debroy, 2023] Xiaojie Zhang and Saptarshi Debroy. Resource management in mobile edge computing: a comprehensive survey. *ACM Computing Surveys*, 55(13):1–37, July 2023.

[Zhang *et al.*, 2023] Haibo Zhang, Xiangyu Liu, Yongjun Xu, Dong Li, Chau Yuen, and Qing Xue. Partial offloading and resource allocation for mec-assisted vehicular networks. *IEEE Transactions on Vehicular Technology*, 73(1):1276–1288, August 2023.

[Zhang *et al.*, 2024] Hangyu Zhang, Hongbo Zhao, Rongke Liu, Aryan Kaushik, Xiangqiang Gao, and Shenzhan Xu. Collaborative task offloading optimization for satellite mobile edge computing using multi-agent deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 73(10):4887–4903, October 2024.