

An Integrated Communication and Computing Scheme for Wi-Fi Networks based on Generative AI and Reinforcement Learning

Xinyang Du, Xuming Fang*

Key Laboratory of Information Coding and Transmission, Southwest Jiaotong University, Chengdu, China
xydu@my.swjtu.edu.cn, xmfang@swjtu.edu.cn

Abstract—The continuous evolution of future mobile communication systems is heading towards the integration of communication and computing, with Mobile Edge Computing (MEC) emerging as a crucial means of implementing Artificial Intelligence (AI) computation. MEC could enhance the computational performance of wireless edge networks by offloading computing-intensive tasks to MEC servers. However, in edge computing scenarios, the sparse sample problem may lead to high costs of time-consuming model training. This paper proposes an MEC offloading decision and resource allocation solution that combines generative AI and deep reinforcement learning (DRL) for the communication-computing integration scenario in the 802.11ax Wi-Fi network. Initially, the optimal offloading policy is determined by the joint use of the Generative Diffusion Model (GDM) and the Twin Delayed DDPG (TD3) algorithm. Subsequently, resource allocation is accomplished by using the Hungarian algorithm. Simulation results demonstrate that the introduction of Generative AI significantly reduces model training costs, and the proposed solution exhibits significant reductions in system task processing latency and total energy consumption costs.

Index Terms—mobile edge computing, reinforcement learning, generative artificial intelligence, computing-communication integration, offloading decision

I. INTRODUCTION

With the rapid proliferation of Internet of Things (IoT) technology, latency-sensitive applications such as autonomous driving, video streaming analysis, virtual reality (AR/VR), and online gaming continue to emerge. There is an increasingly urgent demand for computing services with high speed, low latency, and low energy consumption. Meanwhile, the exponential growth of smart devices rapidly connecting to networks generates a large amount of data. Traditional cloud computing architectures are currently unable to achieve large-scale real-time computing for massive front-end devices. In order to significantly enhance the data processing of wireless edge networks and meet the demand of users for computing service quality, edge computing technology has emerged.

Mobile Edge Computing (MEC) enhances the computational performance of wireless edge networks by offloading locally intensive tasks to MEC servers [1]. As a pivotal technology for integrating communication and computing, MEC meets critical requirements in various areas such as

real-time operations, data optimization, security, and privacy protection. Hence, it finds widespread application in scenarios including vehicular networks, unmanned aerial vehicles, smart cities, and virtual reality services.

In recent years, people have devoted significant research efforts to the joint optimization of task offloading policies and resource allocation. The work in [2] proposed a distributed algorithm based on game theory, which jointly optimizes latency and energy consumption. An MEC system architecture with multiple access points (APs) and STAs was constructed based on the 802.11ac standard in [3]. The work proposes a scheme combining computing offloading and resource allocation. The optimization problem was formulated as an integer programming problem and solved using branch and bound method. However, the above studies all assume that communication resources can be arbitrarily partitioned, which is not applicable to existing Wi-Fi networks. Additionally, due to the complexity and variability of the transmission environment, the actual wireless channel exhibits time-varying characteristics. The solutions mentioned above become impractical. As a branch of artificial intelligence (AI), reinforcement learning (RL) has made significant contributions to decision-making problems in the MEC field. The work in [4] introduced a resource allocation scheme based on state-action-reward-state-action (SARSA) algorithm to solve resource management problems in MEC systems, aiming to minimize the weighted sum of energy consumption and latency. However, there is an issue of low sample efficiency in edge computing scenarios. Due to the lack of expert datasets, deep reinforcement learning (DRL) model typically requires extensive interaction with the environment, leading to high computational costs and time consumption [5].

To tackle this problem, the Generative Diffusion Model (GDM) was proposed in [6]. As a generative AI technique, it has the ability to capture complex data distributions and can seamlessly integrate with other RL policies to reduce the number of samples required and enhance RL performance [7]. GDM utilizes a denoising network to iteratively converge to an approximation of the true sample through a series of estimation steps [8]. After obtaining the initial input, GDM gradually introduces Gaussian noise through a forward diffusion process. Subsequently, a neural network is trained

The work of X. Du, and X. Fang was supported in part by the NSFC under Grant No. 62071393. (Corresponding author: Xuming Fang.)

Based on the mapping between the receiver Received Signal Strength Indicator (RSSI) and Modulation and Coding Scheme (MCS) given in the protocol, we can calculate the noise power and derive the relationship between SNR and MCS. Therefore, based on SNR_l , the size of allocated RU specifications r_l , the mapping function between RU and transmission rate and the mapping function between SNR and the maximum selectable MCS, the maximum achievable uplink transmission rate of STA_l can be expressed as:

$$V_l = V(r_l, \text{MCS}(SNR_l)) \quad (2)$$

C. Computing Model

There are two modes of task execution: local computing and offloading computing. In this paper, it is assumed that computing tasks cannot be divided and can only be offloaded as a whole. When computing locally, it is assumed that the CPU computing capacity of the local device is f_{local} , and the computing demand of the STA_m is c_m . Therefore, the delay and energy consumption of local computing generated by STA_m can be respectively represented as:

$$T_m^{local} = \frac{c_m}{f_{local}} \quad (3)$$

$$E_m^{local} = 10^{-27} \times (f_{local})^2 \times c_m \quad (4)$$

During offloaded computing, there are three steps: 1) the STA uploads the task; 2) the AP performs the computing; and 3) the AP transmits the result back. Since the size of the result data is much smaller compared to the task data [10], the delay and energy consumption of this part are negligible. Assuming that the AP allocates computing resources f_m to STA_m , the data transmission rate of STA_m is V_m , the amount of data for the task is d_m , and the transmission power of STA_m is p_m , the delay and energy consumption of offloaded computation generated by STA_m can be expressed as:

$$T_m^{off} = \frac{c_m}{f_m} + \frac{d_m}{V_m} \quad (5)$$

$$E_m^{off} = p_m \frac{d_m}{V_m} \quad (6)$$

Similarly, the delay and energy consumption generated by communication STA_n can be expressed as follows:

$$T_n^{trans} = \frac{d_n}{V_n} \quad (7)$$

$$E_n^{trans} = p_n \frac{d_n}{V_n} \quad (8)$$

The calculation of total delay and energy consumption can be summarized as follows:

$$T_{total} = \sum_{m=1}^M a_m T_m^{off} + (1 - a_m) T_m^{local} + \sum_{n=1}^N T_n^{trans} \quad (9)$$

$$E_{total} = \sum_{m=1}^M a_m E_m^{off} + (1 - a_m) E_m^{local} + \sum_{n=1}^N E_n^{trans} \quad (10)$$

D. Problem Formulation

Let the decision variable of STA_m be $a_m \in \{0, 1\}$, where $a_m = 0$ represents local execution and $a_m = 1$ represents offloaded execution. Thus, the offloading decision vector can be represented as $\mathcal{A} = \{a_1, a_2, a_3, \dots, a_M\}$. The AP communication resource allocation vector is denoted as $\mathcal{R} = \{r_1, r_2, \dots, r_L\}$, and the computation resource allocation vector is denoted as $\mathcal{F} = \{f_1, f_2, f_3, \dots, f_M\}$. Consistent with [11] and [12], the objective of this paper is to minimize the weighted sum of system delay and energy consumption. Therefore, the optimization problem can be expressed as:

$$\begin{aligned} \mathbf{P}: \min_{\mathcal{A}, \mathcal{R}, \mathcal{F}} & (\lambda T_{total} + (1 - \lambda) E_{total}) \\ \text{s.t. } C1: & (1 - a_m) T_m^{local} + a_m T_m^{off} \leq \tau_m, \forall m \in \mathcal{M} \\ C2: & T_n^{trans} \leq \tau_n, \forall n \in \mathcal{N} \\ C3: & a_m \in \{0, 1\}, \forall m \in \mathcal{M} \\ C4: & \sum_{l \in \mathcal{L}} r_l b_c \leq B^{max} \\ C5: & r_l \in \{1, 2, 4, 9, 18, 36\}, \forall l \in \mathcal{L} \\ C6: & \sum_{m \in \mathcal{M}} f_m \leq F^{max} \end{aligned} \quad (11)$$

where B^{max} and F^{max} respectively denote the maximum communication and computational resources at the AP side, while λ represents the weighting coefficient of the objective. $C1$ indicates the constraint on the maximum delay for computational tasks, $C2$ represents the constraint on the maximum delay for communication tasks, $C3$ ensures that the offloading decision is a binary variable, $C4$ represents the constraint on AP communication resources, $C5$ indicates the constraint on the allocation of RU specifications, and $C6$ denotes the constraint on AP computation resources.

The optimization problem \mathbf{P} involves multiple variables, and the presence of integer variables a_m and r_l makes it a non-convex integer programming problem [13], which has been proven to be NP-Hard in [14], making it difficult to solve using traditional optimization algorithms. Therefore, we formulate \mathbf{P} as a Markov Decision Processes (MDP) and utilize generative AI and DRL to find the optimal values for offloading and resource allocation.

III. OFFLOADING AND RESOURCE ALLOCATION SCHEME BASED ON DTD3

A. Three Key Elements for RL

The key to solving RL problems lies in constructing the agent and environment, which involves designing parameters such as state, action, and reward.

- **State:** We define the system state space as $\mathcal{S} = \{d_1, d_2, d_3, \dots, d_L, c_1, c_2, c_3, \dots, c_M, f_{mec}\}$, which consists of the size of the task data, the number of CPU cycles required for computing, and the available computing resources in MEC.
- **Action:** Considering that the offloading policy in the system is binary, we define the action space as $\mathcal{A} = \{a_1, a_2, a_3, \dots, a_M\}$, where $a_m \in \{0, 1\}$.
- **Reward:** Considering minimizing the weighted sum of latency and energy consumption while prioritizing communication tasks to meet latency constraints, we assume

c_{local} represents the total cost of local computing, c_{total} denotes the total cost of computing in the current time slot, t_n^{trans} stands for the transmission time of communication STA_n , and τ_n represents the latency constraint of communication task. Then we can define the reward function as:

$$\mathcal{R} = \begin{cases} \frac{c_{local} - c_{total}}{c_{local}} - 1, & \forall t_n^{trans} \leq \tau_n \\ -1, & \exists t_n^{trans} > \tau_n \end{cases} \quad (12)$$

B. DTD3 Offloading Decision Algorithm

Considering the high training cost and low efficiency of traditional RL algorithms in edge computing scenarios, this paper proposes a deep diffusion learning model called DTD3 to jointly solve the offloading decision problem in multi-user edge networks. The algorithm architecture of DTD3 consists of several components, including: policy network, twin critic networks, target policy network, target critic network, and experience replay buffer. Unlike traditional RL algorithms, DTD3 uses a Diffusion Policy (DP) based on a diffusion model as the policy network. Similar to [9], we define $k = \{1, 2, 3, \dots, K\}$ as diffusion timestep, and the RL policy of this solution can be represented as the inverse process of conditional diffusion modeling:

$$\pi_\theta(a|s) = \mathcal{N}(a^K; 0, I) \prod_{k=1}^K p_\theta(a^{k-1}|a^k, s) \quad (13)$$

The final sample of the reverse diffusion chain a^0 is the action provided by the model. Through the reverse diffusion process, the model can effectively capture the dependency between the state and the action. Generally, $p_\theta(a^{k-1}|a^k, s)$ can be modeled as a Gaussian distribution $\mathcal{N}(a^{k-1}; \mu_\theta(a^k, s, k), \Sigma_\theta(a^k, s, k))$ and can be parameterized as a noise prediction model, where $\alpha_k = 1 - \beta_k$, $\bar{\alpha}_k = \prod_{i=1}^k \alpha_i$, the covariance matrix is constructed as $\Sigma_\theta(a^k, s, k) = \beta_k I$ and mean is represented as:

$$\mu_\theta(a^k, s, k) = \frac{1}{\sqrt{\alpha_k}} \left(a^k - \frac{\beta_k}{\sqrt{1 - \bar{\alpha}_k}} \varepsilon_\theta(a^k, s, k) \right) \quad (14)$$

The reverse diffusion process first samples $a^K \sim \mathcal{N}(0, I)$, and then multiple steps of sampling for $k = K, \dots, 1$ are performed using the reverse diffusion chain parameterized by θ :

$$a^{k-1}|a^k = \frac{a^k}{\sqrt{\alpha_k}} - \frac{\beta_k}{\sqrt{\alpha_k(1 - \bar{\alpha}_k)}} \varepsilon_\theta(a^k, s, k) + \sqrt{\beta_k} \varepsilon \quad (15)$$

where $\varepsilon \sim \mathcal{N}(0, I)$. According to [15], this approach defines the diffusion model loss as:

$$\mathcal{L}_d(\theta) = \mathbb{E}_{\varepsilon \sim \mathcal{N}(0, I)} \left[\|\varepsilon - \varepsilon_\theta \left(\sqrt{\bar{\alpha}_k} a^0 + \sqrt{1 - \bar{\alpha}_k} \varepsilon, s, k \right)\|^2 \right] \quad (16)$$

Consistent with [9], the Q-value function is injected into the reverse diffusion chain during training model:

$$\mathcal{L}_q(\theta) = -\frac{\eta}{\mathbb{E}_{(s,a) \sim D} [\|Q_\phi(s, a)\|]} \cdot \mathbb{E}_{s \sim D, a^0 \sim \pi_\theta} [Q_\phi(s, a^0)] \quad (17)$$

where η is a hyperparameter used to balance the regularization and the ability of Q-learning. Thus, the final policy function learning objective can be represented as:

$$\pi = \arg_{\pi_\theta} \min(\mathcal{L}_d(\theta) + \mathcal{L}_q(\theta)) \quad (18)$$

The training process of DTD3 is presented in algorithm 1.

Algorithm 1 The training process of Diffusion TD3

Initialize policy network π_θ , twin critic networks Q_{ϕ_1}, Q_{ϕ_2} and target network $\pi'_\theta, Q'_{\phi_1}, Q'_{\phi_2}$

for $t = 1, 2, \dots, T$ **do**

Select action by $a_t \sim \pi_\theta(a_t | s_t)$

Observe r_t, s_{t+1} and store transition (s_t, a_t, r_t, s_{t+1})

Sample mini-batch $\mathcal{B} = (s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}$

$y \leftarrow r(s_t, a_t) + \gamma \min_{i=1,2} Q_{\phi'_i}(s_{t+1}, a_{t+1})$

Update Q_{ϕ_1} and $Q_{\phi_2} \leftarrow \mathbb{E}_{a'_t \sim \pi_{\theta'}} [\|y - Q_{\phi_i}(s_t, a'_t)\|^2]$

Update policy network according to (18)

if $t \bmod d$ **then**

Soft update target network: $\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$

$\phi'_i \leftarrow \tau\phi_i + (1 - \tau)\phi'_i$, for $i \in \{1, 2\}$

end if

end for

C. Resource Allocation Scheme

After completing the offloading decision, the system proceeds with the allocation of computational resources to the computing STAs. We take into account the factors such as the required CPU computational resources c_m , task data size d_m , task latency constraint τ_m and the transmission capacity of STA_m $capability_m$ to calculate the priority of tasks. When an STA has poorer channel condition, larger task data size, greater CPU computational resource requirement, or stricter latency constraint, the task demand is more challenging to fulfill. Therefore, the system should assign a higher priority to such STAs and allocate more computational resources to them. We set $RU_m^{req} = \frac{d_m}{\tau_m}$, $f_m^{req} = \frac{c_m}{\tau_m}$. $capability_m$ represents the maximum MCS achievable under the current environment's channel conditions. Since the communication resources have not been allocated yet, we set r_m for all STAs to 1, which means the RU specification is 26-tone. We let w_1, w_2 and w_3 be weight parameters, and $\sum_{i=1}^3 w_i = 1$. Thus, the priority calculation can be expressed as follows:

$$priority_m = w_1 \frac{RU_m^{req}}{\sum_{m \in \mathcal{M}} RU_m^{req}} + w_2 \frac{f_m^{req}}{\sum_{m \in \mathcal{M}} f_m^{req}} + w_3 \frac{\frac{1}{capability_m}}{\sum_{m \in \mathcal{M}} \frac{1}{capability_m}} \quad (19)$$

The computational resource allocation is represented as:

$$f_m = \frac{priority_m}{\sum_{m=1}^M priority_m} \times f_{mec} \quad (20)$$

Based on the resource allocation characteristics of 802.11ax OFDMA, we consider the allocation problem of communication resources as an allocation problem of RU specifications. To address this issue, this paper first determines the combination of RU specifications used based on the

number of transmission tasks of the AP. If the number of transmission tasks of the AP exceeds the maximum number of RUs that the AP can allocate, the tasks are sorted based on the calculated task priorities. RUs are allocated to tasks with higher priority, and resource redistribution is performed according to the computing resource allocation scheme. Tasks with lower priority are no longer offloaded. Given the limited bandwidth resources, the number of partitioning methods for RU specifications is limited. First, we obtain possible RU specification combinations based on the number of STAs and available bandwidth. Then, we calculate the efficiency matrix for each RU configuration combination and use the Hungarian algorithm to determine the optimal RU allocation scheme along with the corresponding STA transmission latency and energy consumption. Finally, we select the RU configuration combination that minimizes STA transmission latency and energy consumption as the final RU allocation result.

IV. SIMULATION RESULT

We use Python 3.7 and PyTorch to build the simulation platform and conduct algorithm training and simulation. The algorithm model is deployed on the AP for centralized training in this paper. Assuming the AP's position keeps fixed, STAs are randomly distributed within a circle around the AP with the radius of 20m. All STAs complete computing tasks or communication tasks transmission via OFDMA. We modeled indoor path loss using the Keenan-Motley model. The computing task latency limit for STAs is set to 80%-120% of the latency when the task is computed locally. The computational capacities of MEC and STA are 10GHz and 1GHz, respectively. The transmission power of STAs is 500mW. The number of CPU cycles required for computing is uniformly distributed between 900 Megacycles and 1100 Megacycles [14]. The data size of computing task is uniformly distributed between 2.4Mbits and 4Mbits, and the communication task is between 10Mbits and 20Mbits. λ is set to 0.8.

To validate the superiority of our algorithm, we compare the performance of the proposed DTD3 scheme with four baseline schemes through simulations. "Local computing" means that all computing tasks are executed locally, "Full offloading" stands for that all computing tasks are offloaded, and "Random offloading" stands for that AP randomly determines the offloading decision. The above three baselines allocate RUs evenly among STAs. Further, "DQN" stands for that the offloading decision is determined by DQN model, and the RU is allocated using the Hungarian algorithm. The Quality of Service (QoS) represents the ratio of computing tasks satisfying the latency limit to the total number of computing tasks, while the communication success rate represents the ratio of communication tasks completed within the latency limit to the total number of communication tasks. We vary some parameters in simulation to observe the performance.

A. Scenario with Varying Computing STAs

Fig. 2 illustrates the performance of algorithms with varying computing STAs when there are 3 communication STAs. In

Fig. 2a, the total cost increases with the increase in computing STAs. DTD3 demonstrates outstanding performance among all approaches. Fig. 2b indicates that due to the limited computing and communication resources at the AP, the QoS decreases as the number of computing STAs increases. However, under resource constraints, the QoS of DTD3 is significantly higher than other approaches. Fig. 2c shows that even with an increasing number of computing STAs, the communication success rate of DTD3 remains at a high level. Similarly, the trained DQN model can also provide offloading decisions that meet the communication STAs' requirements. In contrast, as the resources of the AP cannot meet the demands of all STAs, the communication success rate of the full offloading scheme significantly decreases with the increase of computing STAs.

B. Scenario with Varying Capacity of MEC

Fig. 3 depicts the performance of algorithms with varying capacity of MEC when there are 3 communication STAs and 5 computing STAs. Fig. 3a shows that as the MEC computing resource increases, the total cost decreases due to the corresponding reduction in system latency and energy consumption. In contrast, the performance trend of the local computing policy remains relatively stable as it is independent of MEC computing resources. DTD3 significantly outperforms other approaches. Fig. 3b illustrates that the QoS of computing STAs increases with the increase of the MEC computing resource. DTD3 achieves higher QoS compared to other approaches, particularly when MEC computing resources are limited. Fig. 3c shows that the communication success rate of the full offloading policy maintains around 50%, while the random offloading policy maintains around 85%. In contrast, the communication success rate of DTD3 and the DQN algorithm consistently maintains at 100%.

C. Convergence Analysis

Fig. 4 shows the convergence performance of DQN, SAC and the proposed DTD3 algorithms when there are 3 communication STAs, 10 computing STAs, and 10GHz of MEC resources. The DTD3 algorithm converges around 400 episodes with optimal convergence performance. This is attributed to the introduced generative diffusion model, which effectively reduces the convergence time and training cost by working in coordination with the RL framework. In contrast, the convergence and stability of the DQN algorithm are relatively poor. The SAC algorithm, on the other hand, exhibits better convergence performance than the DQN algorithm due to the introduction of the maximum entropy mechanism, which enhances SAC's exploration capability and robustness.

V. CONCLUSION

In this paper, we propose an offloading decision and resource allocation scheme based on generative AI and DRL for the integration of communication and computing in multi-STAs single-AP scenarios under 802.11ax Wi-Fi networks. We introduce diffusion models to address the sparse sample problem in edge computing scenarios and propose a

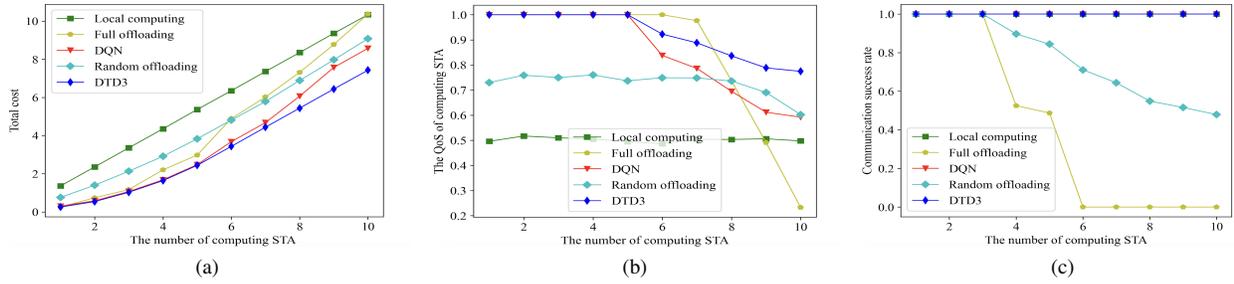


Fig. 2. The performance of algorithms with varying computing STAs: (a) total cost versus the number of computing STAs, (b) QoS versus the number of computing STAs, (c) communication success rate versus the number of computing STAs

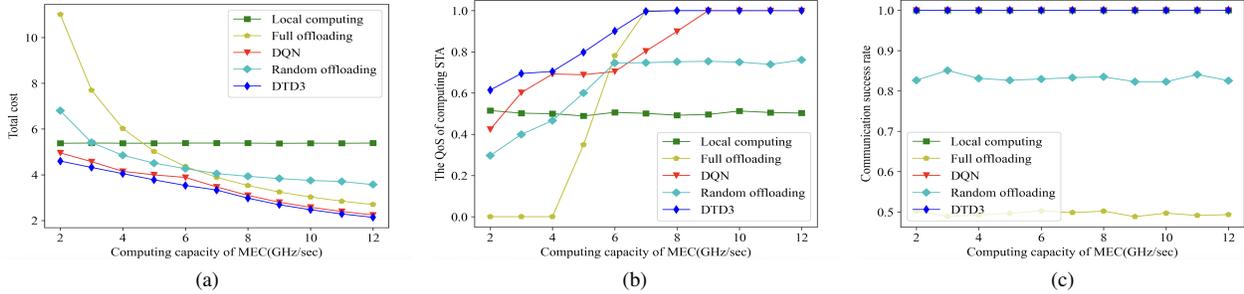


Fig. 3. The performance of algorithms with varying capacity of MEC: (a) total cost versus the capacity of MEC, (b) QoS versus the capacity of MEC, (c) communication success rate versus the capacity of MEC

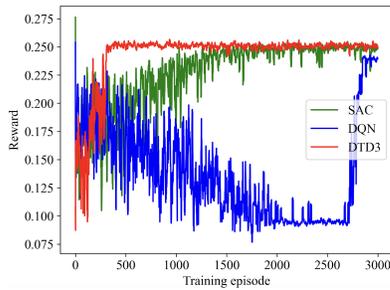


Fig. 4. The convergence of algorithms

communication allocation scheme more suitable for Wi-Fi environments based on the Hungarian algorithm. Extensive simulation results demonstrate that the proposed approach can reduce the overall energy consumption and latency of the system, enhance QoS, and ensure communication success rate. Moreover, compared to traditional RL methods, this model exhibits superior convergence performance. In the future, our framework could be extended to multi-agent mobile edge systems, utilizing distributed execution of RL algorithms to improve system scalability and robustness.

REFERENCES

- [1] M. Tang and V. W. S. Wong, "Deep Reinforcement Learning for Task Offloading in Mobile Edge Computing Systems," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 1985-1997, 2022.
- [2] L. Yang, H. Zhang, X. Li, H. Ji, and V. C. M. Leung, "A Distributed Computation Offloading Strategy in Small-Cell Networks Integrated With Mobile Edge Computing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2762-2773, 2018.
- [3] B. Dab, N. Aitsaadi, and R. Langar, "Joint Optimization Of Offloading And Resource Allocation Scheme For Mobile Edge Computing," in *Proc. IEEE WCNC*, 2019, pp. 1-7.
- [4] T. Alfakih, M. M. Hassan, A. Gumaie, C. Savaglio, and G. Fortino, "Task Offloading and Resource Allocation for Mobile Edge Computing by Deep Reinforcement Learning Based on SARSA," *IEEE Access*, vol. 8, pp. 54074-54084, 2020.
- [5] N. C. Luong, S. Gong, D. Niyato, P. Wang, Y. Liang, and D. I. Kim, "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133-3174, 2019.
- [6] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep Unsupervised Learning using Nonequilibrium Thermodynamics," in *Proc. ICML*, 2015, pp. 2256-2265.
- [7] H. Du, R. Zhang, Y. Liu, J. Wang, Y. Lin, Z. Li, D. Niyato, J. Kang, Z. Xiong, S. Cui, A. Bo, H. Zhou, I. Dong, et al., "Beyond Deep Reinforcement Learning: A Tutorial on Generative Diffusion Models in Network Optimization," *CoRR*, vol. abs/2308.05384, 2023.
- [8] H. Du, R. Zhang, D. Niyato, J. Kang, Z. Xiong, D. I. Kim, X. Shen, and H. V. Poor, "Exploring Collaborative Distributed Diffusion-Based AI-Generated Content (AIGC) in Wireless Networks," *CoRR*, vol. abs/2304.03446.99, pp. 1-8, 2023.
- [9] Z. Wang, J. Huang, and M. Zhou, "Diffusion Policies as an Expressive Policy Class for Offline Reinforcement Learning," in *Proc. ICLR*, 2022.
- [10] P. Zhao, H. Tian, C. Qin, and G. Nie, "Energy-Saving Offloading by Jointly Allocating Radio and Computational Resources for Mobile Edge Computing," *IEEE Access*, vol. 5, pp. 11255-11268, 2017.
- [11] W. Zhan, C. Luo, J. Wang, G. Min, and H. Duan, "Deep Reinforcement Learning-Based Computation Offloading in Vehicular Edge Computing," in *Proc. GLOBECOM*, 2019, pp. 1-6.
- [12] H. Cao and J. Cai, "Distributed multiuser computation offloading for cloudlet-based mobile cloud computing: A game-theoretic machine learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 752-764, 2018.
- [13] N. Eshraghi and B. Liang, "Joint Offloading Decision and Resource Allocation with Uncertain Task Computing Requirement," in *Proc. IEEE INFOCOM*, Paris, France, Apr. 2019, pp. 1414-1422.
- [14] J. Li, H. Gao, T. Lv, Y. Lu, "Deep Reinforcement Learning Based Computation Offloading and Resource Allocation for MEC," in *Proc. IEEE WCNC*, 2018, pp. 1-6.
- [15] J. Ho, A. Jain, and P. Abbeel, "Denosing Diffusion Probabilistic Models," in *Proc. NeurIPS*, 2020, vol. 33, pp. 6840-6851.