

APPENDIX A DIFFERENCES IN GENERALIZATION COMPARED TO RELATED WORKS

We summarize the distinctions between our approach and existing works. Many studies [1]–[6] are limited to problems that optimize for a single preference. Li et al. [3] employ a Q-learning-based deep reinforcement learning (DRL) method to solve the computation offloading problem in a multi-user environment. Cui et al. [1] decomposes user association, offloading decision, computing, and communication resource allocation into two related sub-problems and employs the DQN algorithm for decision-making. Lei et al. [2] proposed a DRL-based joint computation offloading and multi-user scheduling algorithm for IoT edge computing systems, aiming to minimize the long-term weighted sum of delay and power consumption under stochastic traffic arrivals. Huang et al. [4] employed an improved DQN method to address offloading decision problems and resource allocation problems. The above works focus on two objectives, delay and energy consumption, and use a weight coefficient to balance them or optimize one objective while satisfying the constraints of the other. Moreover, these studies lack research on the generalization.

Some studies [7]–[11] focus only on the generalization of system parameters. Li et al. [7] combine graph neural networks and seq2seq networks to make decisions on task offloading. They employ a meta-reinforcement learning approach to enhance the generalization of the offloading strategy in environments with different system parameters. Ren et al. [8] design a set of experience maintaining and sampling strategies to improve the training process of DRL, enhancing the model's generalization to different environments. Wang et al. [9] design an offloading decision algorithm based on meta-reinforcement learning, which uses a seq2seq neural network to represent the offloading policy. This approach can adapt to various environments covering a wide range of topologies, task numbers, and transmission rates. Wu et al. [10] propose a method that combines graph neural networks and DRL, which can be applied to various environments with inter-dependencies among different tasks. Hu et al. [11] propose a size-adaptive offloading scheme and a setting-adaptive offloading component, designed to quickly adapt to new MEC environments of varying sizes and configurations with a few interaction steps. The above work only considers generalization in terms of system parameters, without addressing generalization in terms of the number of servers and multi-preference issues.

Other works [12]–[14] only consider the generalization of the number of servers. A few works consider the generalization of both system parameters and the number of servers. Gao et al. [15] model the decentralized task offloading problem as a partially observable Markov decision process and use a multi-agent RL method to train the policy. They consider the generalization of both system parameters and the number of servers, but do not explore multi-preference issues. Our method provides a deeper exploration of the generalization of the offloading strategy, considering the generalization in terms of multi-preference, system parameters, and server quantities.

APPENDIX B SUPPLEMENTARY FIGURES

B.1 System Model

The MEC system model we consider is illustrated in Fig. A1. An MEC system consists of E edge servers, one remote cloud server. The system processes M tasks arriving sequentially, with each task being uploaded to only one server.

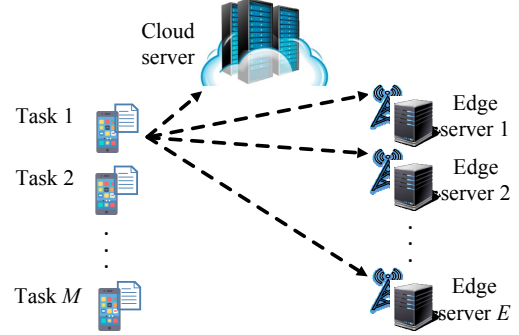


Fig. A1: An illustrative example system model of MEC.

B.2 Learning Approach

During the training phase, we sample N_g contexts to create N_g MEC environments for each epoch. The preferences of these environments are determined by Eq. (32), while their number of servers E and frequencies f_E are randomly drawn from the context space. These environments interact with the policy to generate experiences, which are stored in the replay buffer and used to update the policy.

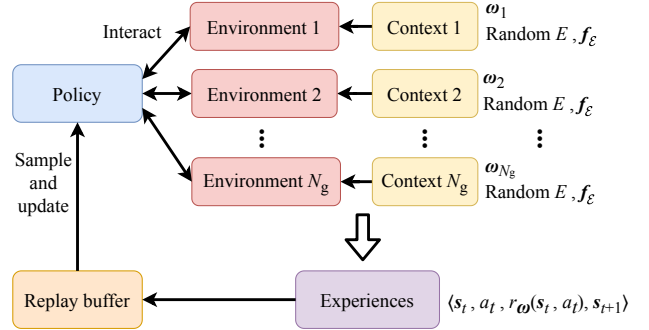


Fig. A2: The generalization learning approach.

APPENDIX C SIMULATION SETUP

We provide the context in Table A1. We set testing preference set Ω_{N_g} according to Eq. (32) and fit Pareto front in N_g preferences. Each preference's performance contains total delay and energy consumption for all tasks in one episode. We evaluate a performance (delay or energy consumption) with an average of 1000 episodes. A disk coverage has a radius of 1000m to 2000m for a cloud server and 50m to 500m for an edge server. Each episode needs to initial different radiuses for the cloud and edge servers. We set the mean of task size \bar{L} according to Eq. (1).

TABLE A1
CONTEXT SPACE FOR TRAINING AND TESTING

Context space	Training	Testing
The number of preference N_g	64	101
Edge server quantity C_E	$\{1, 2, \dots, 8\}$	$\{1, 2, \dots, 10\}$
Cloud server CPU frequency C_{f_0}	$[3.5, 4.5]$ GHz	$[3.0, 5.0]$ GHz
Edge server CPU frequency $C_{f_{e'}}$	$[1.75, 2.25]$ GHz	$[1.5, 2.5]$ GHz

C.1 Evaluation Metrics

We consider the following metrics to evaluate the performances of the proposed algorithms.

- **Energy Consumption:** The total energy consumption of one episode given as $\sum_{m=1}^M E_m^{\text{off}} + E_m^{\text{exe}}$, and the average energy consumption per Mbits task of one episode given by $\sum_{m=1}^M \frac{E_m^{\text{off}} + E_m^{\text{exe}}}{L}$.
- **Task Delay:** The total energy consumption of one episode given as $\sum_{m=1}^M E_m^{\text{off}} + E_m^{\text{exe}}$, and the average energy consumption per Mbits task of one episode given by $\sum_{m=1}^M \frac{E_m^{\text{off}} + E_m^{\text{exe}}}{L}$.
- **Pareto Front:**
 $PF(\Pi) = \{\pi \in \Pi \mid \nexists \pi' \in \Pi : \mathbf{y}^{\pi'} \succ_P \mathbf{y}^{\pi}\}$, where the symbols are defined by Eq. (12).
- **Hypervolume Metric:**
 $\mathcal{V}(PF(\Pi)) = \int_{\mathbb{R}^2} \mathbb{I}_{V_h(PF(\Pi))}(z) dz$, where the symbols are defined by Eq. (14).

C.2 Baselines

LinUCB-based scheme: The Offloading scheme is based on a kind of contextual MAB algorithm [16]. It is an improvement over the traditional UCB algorithm. This scheme uses states as MAB contexts and learns a policy by exploring different actions. We apply the multi-arm bandit algorithm. We regard each action as an arm and construct the feature of an arm from preference ω and server information vector $\mathbf{s}_{t,e}$. Then, we update the parameter matrix based on the context and exploration results to learn a strategy that maximizes rewards. We train this scheme in preference set Ω_{101} and evaluate it for any preference in one. This method is computationally simple and incorporates context information, making it widely used in task offloading.

SA-based scheme: The heuristic method searches for an optimal local solution for task offloading without contexts. We use this method to observe the performance of heuristic approaches. This method generates a fixed offloading scheme for each preference and then iteratively searches for better solutions through local search. Once a better solution is

found, it is accepted or rejected with a certain probability. This scheme searches 10000 episodes for each preference. However, searching for a solution that only applies to a specific context is time-consuming.

Random-based scheme: The random-based scheme has p probability to offload a task to the cloud server and $1 - p$ probability to a random edge server. We tune the probability p and evaluate the scheme to obtain a Pareto front.

Multi-policy scheme: The multi-policy MORL approach [13] is based on the standard Discrete-SAC algorithm. We build 101 Discrete-SAC policy models for the 101 preference in Ω_{101} correspondingly. We train each policy model with $f_0 = 4$ GHz and $f_{e'} = 2$ GHz. This method has no generalization ability. A well-trained policy model is applicable to a specific context. However, benefiting from focusing on a specific context, this method is more likely to achieve optimal performance. We apply the method to determine the upper bound of the Pareto front.

C.3 Convergence Performances

We verify the convergence of the proposed GMORL algorithm. In Fig. A3a, we evaluate and plot the training reward of our algorithm. The reward shown in this figure is scalarized using Eq. (29). We observe that with the training episode increasing, the total reward converges. In fig. A3b and fig. A3c, as the training episodes increase, the delay and energy consumption decrease and converge to a stable value. This indicates that the GMORL algorithm converges effectively and reach a Pareto local optimum. In the following subsection, we will specifically analyze other performances in various system settings.

REFERENCES

- [1] G. Cui, X. Li, L. Xu, and W. Wang, "Latency and energy optimization for mec enhanced sat-iot networks," *IEEE Access*, vol. 8, pp. 55915–55926, 2020.
- [2] L. Lei, H. Xu, X. Xiong, K. Zheng, W. Xiang, and X. Wang, "Multiuser resource control with deep reinforcement learning in iot edge computing," *IEEE Internet of Things J.*, vol. 6, no. 6, pp. 10119–10133, 2019.
- [3] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for mec," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2018, pp. 1–6.
- [4] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581–2593, 2019.
- [5] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Deep reinforcement learning for collaborative offloading in heterogeneous edge networks," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 2021, pp. 297–303.
- [6] F. Jiang, L. Dong, K. Wang, K. Yang, and C. Pan, "Distributed resource scheduling for large-scale mec systems: A multiagent ensemble deep reinforcement learning with imitation acceleration," *IEEE Internet of Things Journal*, vol. 9, no. 9, pp. 6597–6610, 2021.
- [7] Y. Li, J. Li, Z. Lv, H. Li, Y. Wang, and Z. Xu, "Gasto: A fast adaptive graph learning framework for edge computing empowered task offloading," *IEEE Transactions on Network and Service Management*, 2023.
- [8] T. Ren, J. Niu, and Y. Qiu, "Enhancing generalization of computation offloading policies in novel mobile edge computing environments by exploiting experience utility," *Journal of Systems Architecture*, vol. 125, p. 102444, 2022.

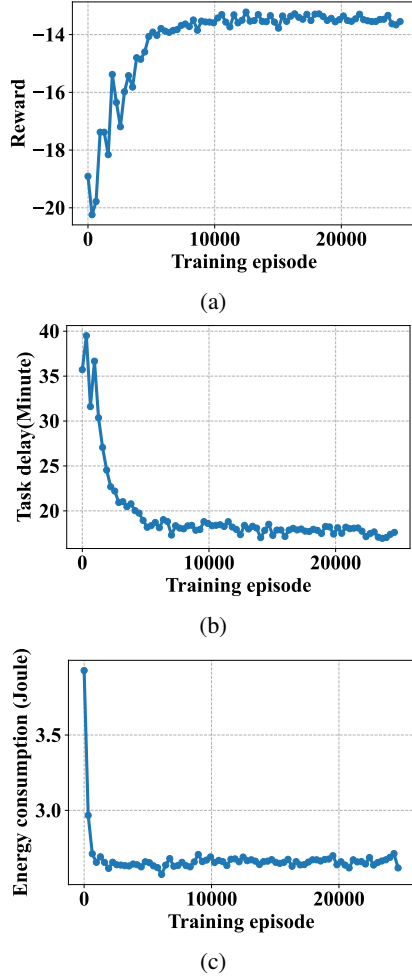


Fig. A3: Convergence performance of the proposed GMORL algorithm: (a) Reward during training; (b) Total delay during training when $E = 5$, $f_0 = 4$ GHz, $f_{e'} = 2$ GHz for all $e' \in \mathcal{E}'$, and $\omega = (1, 0)$; (c) Total energy consumption during training when $E = 5$, CPU frequency $f_0 = 4$ GHz, $f_{e'} = 2$ GHz for all $e' \in \mathcal{E}'$, and preference $\omega = (0, 1)$.

of the 19th international conference on World wide web, 2010, pp. 661–670.

- [9] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, “Fast adaptive task offloading in edge computing based on meta reinforcement learning,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 242–253, 2020.
- [10] T. Wu, W. Jing, X. Wen, Z. Lu, and S. Zhao, “A scalable computation offloading scheme for mec based on graph neural networks,” in *2021 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2021, pp. 1–6.
- [11] Z. Hu, J. Niu, T. Ren, and M. Guizani, “Achieving fast environment adaptation of drl-based computation offloading in mobile edge computing,” *IEEE Transactions on Mobile Computing*, 2023.
- [12] F. Jiang, K. Wang, L. Dong, C. Pan, and K. Yang, “Stacked autoencoder-based deep reinforcement learning for online resource scheduling in large-scale mec networks,” *IEEE Internet of Things J.*, vol. 7, no. 10, pp. 9278–9290, 2020.
- [13] N. Yang, J. Wen, M. Zhang, and M. Tang, “Multi-objective deep reinforcement learning for mobile edge computing,” in *2023 21st international symposium on modeling and optimization in mobile, ad hoc, and wireless networks (WiOpt)*. IEEE, 2023, pp. 1–8.
- [14] J. Chang, J. Wang, B. Li, Y. Zhao, and D. Li, “Attention-based deep reinforcement learning for edge user allocation,” *IEEE Transactions on Network and Service Management*, 2023.
- [15] Z. Gao, L. Yang, and Y. Dai, “Fast adaptive task offloading and resource allocation in large-scale mec systems via multi-agent graph reinforcement learning,” *IEEE Internet of Things Journal*, 2023.
- [16] L. Li, W. Chu, J. Langford, and R. E. Schapire, “A contextual-bandit approach to personalized news article recommendation,” in *Proceedings*