MADDPG with Parameter Sharing for Robust Resource Allocation

Fitsum Debebe Tilahun, and Chung G. Kang
School of Electrical Engineering, Korea University
{fitsum_debebe, and ccgkang} @korea.ac.kr

Abstract

This paper presents an approach to enhance the generalization capability of MADDPG algorithm-based resource allocation scheme, leveraging parameter sharing through a single agent with a shared actor-critic pair. Our approach focuses on improving the algorithm's adaptability to unforeseen scenarios, especially variations in the number of users encountered post-training. The results demonstrate that the pre-trained model serves as a robust foundation, facilitating rapid adaptation to the dynamic nature of the system, and ensuring efficient performance even for users joining the environment after the initial training phase.

I. Introduction

In our earlier work [1], we delved into a joint communication and computing resource allocation (JCCRA) in a cell-free massive MIMO-enabled mobile edge computing (MEC) system, with the primary objective of minimizing the total energy consumption while meeting stringent user delay requirements. To tackle the problem, we presented a distributed JCCRA scheme based on a cooperative multi-agent reinforcement learning (MARL), specifically leveraging the multi-agent deep deterministic policy gradient (MADDPG) algorithm [2] under the framework of centralized training and decentralized execution (CTDE). Thus, each user is treated as a deep reinforcement learning (DRL) agent, equipped with its own actor network for learning an individual policy, while a dedicated critic network provides individualized expected return taking into account the joint actions and states of all agents in the system. Consequently, the algorithm becomes specialized to the specific conditions and dynamics present in the training environment, failing to generalize well to cases when the number of users in the system undergoes changes, such as arrivals and departures.

In this work, to improve generalization of the algorithm, while speeding up the learning process, we implement a single agent, which entails a utilization of a shared actor and critic pair among the agents. This shared agent undergoes training using experiences sampled from the interactions of all agents, stored collectively in a replay buffer. By consolidating the knowledge gained from the diverse experiences of all agents, we seek to create a more robust and adaptable framework that can efficiently handle changes in the number of users within the system.

II. System Model and Problem Formulation

At the beginning of each discrete time step *t*, each user generates a computationally intensive task with $Q_k(t)$ bits to be processed within the application deadline, denoted by t_k^d. Let $f_k^{\max}$ represent the maximum local processing speed of the *k*-th user, while N_{cpb} corresponds to the number of cycles required to process a one-bit task. Furthermore, Let $Q_k^{local}(t) = \alpha_k(t) Q_k(t)$ for $\alpha_k \in [0,1]$ denote the proportion of locally computed task size, implying the task bits to be offloaded is given by $Q_k^{offload}(t) = Q_k(t) - Q_k^{local}(t)$. Then, the time consumed for local execution can be expressed as $t_k^{local}(t) = \min\left(\frac{Q_k^{local}(t) N_{cpb}}{f_k^{\max}}, t_k^d\right)$, while the corresponding energy consumption is given by $E_k^{local}(t) = \varsigma Q_k^{local}(t) N_{cpb} \left(f_k^{\max}\right)^2$, where ς is the effective switched capacitance constant.

The remaining task bits, i.e., $Q_k^{offload}(t)$ are offloaded to the edge server for parallel execution. Let the processing capacity of the edge server be denoted by f^E, which is to be proportionally shared among the users based on the offloaded task sizes. Denoting the allocated computation resource for the *k*-th user by $f_k^E(t)$, then the total delay for edge computation $t_k^E(t)$ can be given as a sum of computing delay $t_k^{comp}(t) = \frac{Q_k^{offload}(t) N_{cpb}}{f_k^E(t)}$, and transmission delay $t_k^{tr}(t) = \frac{Q_k^{offload}(t)}{R_k(t)}$. Moreover, the energy consumption of the *k*-th user for edge computing is given as $E_k^E(t) = p_k(t) t_k^{tr}(t)$, where $p_k(t) = \eta_k(t) p_k^{\max}$ is the transmit power, which is determined by power control factor $\eta_k \in [0,1]$, denoting the maximum uplink transmit power of the *k*-th user by $p_k^{\max}$.

Consequently, the overall delay to process $Q_k(t)$ bits locally and at the edge server can be expressed as $t_k(t) = \max\left(t_k^{local}(t), t_k^{offload}(t)\right)$. Similarly, the corresponding energy consumption is given as $E_k(t) = E_k^{local}(t) + E_k^{offload}(t)$. Thus, the objective of JCCRA problem is to optimize the allocation of local processor speed and uplink transmit power for each user to minimize the total energy consumption, $\sum_{k=1}^{K} E_k(t)$, subject to meeting the delay requirements, i.e., $t_k(t) \le t_k^d(t)$, at each time step *t*.

III. Proposed Approach

Each user is modeled as a DRL agent and trained according to MADDPG algorithm, but implemented as a single agent wherein a shared actor and critic pair is utilized among all agents. , denoted by the parameters θ^{μ} and, θ^{Q} respectively, along with the respective copies which serve as targets. This parameter sharing

mechanism ensures that the actor and critic networks of individual agents share the same set of parameters.

We recall the three key elements of the DRL formulation from [1]. More specifically, the local observation of agent k at time step t is defined as $o_k(t) \triangleq \{Q_k(t), t_k^d, R_k(t-1)\}$, where $R_k(t-1)$ is the uplink rate of user k from previous time step $(t-1)$. Given the local observation $o_k(t)$, the agent determines the JCCRA decision variables $a_k(t) = (\alpha_k(t), \eta_k(t))$, which govern the proportion of task size to be computed locally and at the edge, and uplink transmit power allocations. To enforce a cooperation among the agents, each agent receives a common reward signal $r_k(t) = -\sum_{k=1}^{K} \xi_k E_k(t)$, where $\xi_k = 1$ if the delay constraint of the k-user is fulfilled, i.e., $t_k(t) \le t_k^d$ otherwise $\xi_k = 10$ to punish the agent for a failure in meeting the constraint. The transition of all agents $\langle o_k(t), a_k(t), r_k(t), o_k'(t) \rangle, \forall k$ is stored in the replay buffer.

The shared agent undergoes training using experiences sampled from the interactions of all agents, collectively stored in a replay buffer. This design choice not only aims to promote better generalization across varying scenarios but also to streamline the learning trajectory by consolidating knowledge gained from the diverse experiences of all agents. By employing parameter sharing and leveraging insights from collective experiences, our approach establishes a more robust and adaptable framework capable of efficiently handling changes in the number of users within the system. Accordingly, when a user enters the system, it has the capability to download the pre-trained shared model, enabling it to perform effectively without incurring significant performance loss.

IV. Simulation Results

We consider a cell-free MEC system with $M = 49$ APs which are uniformly distributed on an area of 1km^2. Furthermore, we consider a varying number of users (K) in the system, however fixed to $K = 10$ during training. The large scale channel gain is given as $\beta_{mk} = -30.5 - 36.7 \log_{10}(d_{mk}) + F_{mk}$, where d_{mk} is the distance between the k-th user and m-th AP, while $F_{mk} \sim \mathcal{CN}(0,16)$ corresponds to a shadow fading. The system bandwidth is set to 10MHz. Furthermore, the size of the computational task at each user is assumed to be uniformly distributed in the range of [3, 7] Mbps. Other simulation parameters are set as follows: $\Delta t = t_k^d(t) = t_k^d = 1ms$, $N_{cpb} = 500$, $f_k = 1GHz$, $f_k^E = 100GHz$. Unless stated otherwise, other simulation parameters are consistent with the one in [1].

The shared actor and critic pair is implemented as a fully connected networks, with a single hidden layer of 512 neurons. The respective hidden layers are activated using ReLU function, while the output of the actor network is activated using sigmoid. The learning rate of the actor is set to 0.0001, while the critic learning rate is fixed to 0.001. Further, the target update parameter is set to $\tau = 0.001$.

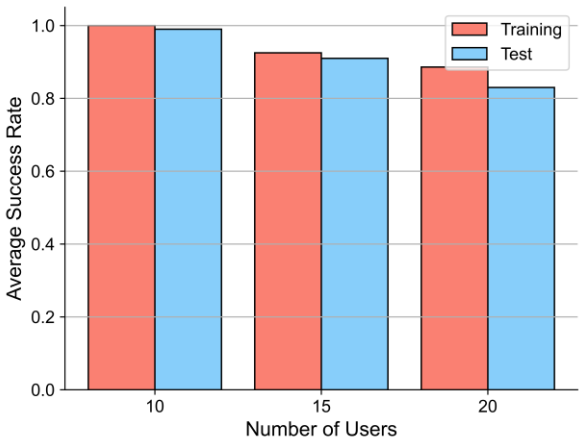

Fig. 1: Performance of the algorithm's generalization for varying number of users in the system

In Fig. 1, we illustrate the average success rate, i.e., successfully meeting the delay constraints, of the proposed algorithm trained at $K = 10$ as the baseline. Subsequently, the algorithm's performance is assessed for various user scenarios, specifically K = 10, 15, and 20. To provide a comprehensive comparison, we introduce a shared actor trained at $K = 15$ and $K = 20$. As depicted in Fig. 1, the proposed algorithm has scaled well and effectively adapts to unseen number of users at the baseline, closely aligning its performance with individually trained models for $K = 15$ and $K = 20$.

V. Conclusions

We enhanced the vanilla MADDPG algorithm by introducing consolidated parameter sharing through a single agent with a shared actor-critic pair, aiming to improve the algorithm's generalization capabilities, allowing it to adapt seamlessly to unforeseen scenarios during training, especially in the presence of a varying number of users.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.2020R1A2C100998413).

References

[1] F. D. Tilahun, A. T. Abebe and C. G. Kang, "Multi-Agent Reinforcement Learning for Distributed Resource Allocation in Cell-Free Massive MIMO-enabled Mobile Edge Computing Network," in *IEEE Transactions on Vehicular Technology*, June 2023.

[2] R. Lowe et al., "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments", arXiv preprint arXiv: 1706.02275, 2020.