Learning-Based Resource Allocation for Heterogeneous Workloads in Cell-Free MEC Network

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

***Abstract*—Cell-free massive MIMO (CF-mMIMO) can enable MEC by providing seamless, high-quality connectivity through macro-diversity. While existing studies often assume uniform task models, this paper investigates a heterogeneous MEC environment where users have class-specific workloads, computation densities, and multi-step deadlines. We propose a learning-based framework for joint resource allocation, optimizing local CPU frequency and uplink transmit power to minimize energy while meeting hard deadlines. Simulations with two traffic classes (XR and IoT) show that applying our framework significantly improves deadline satisfaction and the energy–reliability tradeoff compared to a heuristic baseline.**

***Index Terms*—Cell-free massive MIMO, mobile edge computing, heterogeneous traffic classes, deep reinforcement learning**

I. Introduction

Cell-free massive multiple-input multiple-output (CF-mMIMO) has recently attracted significant attention as a promising architecture for computation offloading in mobile edge computing (MEC) systems, owing to its macro-diversity, uniform service quality, and flexible resource coordination across distributed access points. Leveraging these advantages, a growing body of work has investigated joint communication–computation resource allocation for CF-mMIMO-enabled MEC, including our prior work in [1]. However, most existing studies adopt a *uniform task model*, where task profiles are drawn from the same distribution for all users, i.e., there is no task-based user differentiation in terms of workload, computation density, or latency requirements.

Such uniform assumptions substantially limit practical relevance because real MEC deployments must concurrently support heterogeneous services (e.g., latency-critical XR/AR and industrial IoT) whose task sizes, processing densities, arrival patterns, and deadline requirements differ markedly across users. Moreover, for several applications the deadline naturally spans multiple slot durations, so a task may persist across consecutive decision steps until completion or expiration, introducing temporal correlation through remaining workload and remaining time budget. This multi-step deadline structure makes resource allocation fundamentally more challenging than the commonly assumed per-slot completion model, and it calls for evaluation frameworks and policies that can capture class-specific QoS behavior rather than only averaged outcomes.

II. System Model

We consider a CF-mMIMO-enabled MEC system with M single-antenna access points (APs) jointly serving K single-antenna users (UEs) in the uplink. In this conference version, all APs participate in serving each UE (no clustering). Time is slotted with index $t \in \{1, 2, \ldots\}$ and step duration Δt (set to 1 ms in our simulator). A new UE/AP topology and large-scale fading realization are initialized at the beginning of each episode, while small-scale fading varies across time steps.

Let $g_{mk}(t) = \sqrt{\beta_{mk}}\, h_{mk}(t)$ denote the uplink channel from UE k to AP m, where β_{mk} captures large-scale fading and $h_{mk}(t) \sim \mathcal{CN}(0,1)$ is Rayleigh fading. With maximum-ratio combining across APs, the CPU forms an effective uplink SINR, $\mathrm{SINR}_k(t)$, for UE k (including inter-user interference). The achievable uplink rate is modeled as $R_k(t) = W\left(\frac{\tau_c - \tau_p}{\tau_c}\right)\log_2(1+\gamma_k(t))$, where W is the system bandwidth.

At the beginning of each episode, each UE k is assigned an application class $c_k \in \{1, \ldots, \tilde{C}\}$ (fixed over the episode). Each class c is associated with a task profile distribution over (i) input size T (bits), (ii) computation density C (cycles/bit), and (iii) hard deadline D (seconds). Tasks arrive periodically according to the UE class: if P_{c_k} denotes the class-dependent period (in steps), UE k generates a new task at steps $t \in \{1, 1+P_{c_k}, 1+2P_{c_k}, \ldots\}$, truncated near the episode end. Each arriving task for UE k is characterized by

$$\mathcal{J}_k^n = \left(T_k^n,\ C_k^n,\ D_k^n\right), \tag{1}$$

where n indexes the task arrivals of UE k. The tuple (T_k^n, C_k^n, D_k^n) is drawn from a class-dependent distribution $\mathcal{D}_{c_k}$, enabling heterogeneous workloads and deadlines across users and time. For instance, XR/AR and video analytics typically involve larger input sizes and higher computation intensity with stringent latency constraints, whereas industrial IoT tasks are often lighter but may exhibit different (often less demanding) timing requirements depending on the control/monitoring function.

A task may require multiple time steps to finish. Let $Q_k(t)$ denote the remaining (unprocessed) bits of UE k's *current* task at the beginning of step t, and let $d_k(t)$ denote the remaining deadline budget (seconds) for that task. At each step t, UE k chooses: (i) a local CPU scaling factor $\alpha_k(t) \in [0,1]$ and (ii) a

transmit power factor $\eta_k(t) \in [0,1]$, where $p_k(t) = \eta_k(t) p_k^{\max}$. Only the time window

$$\tau_k(t) = \min\{\Delta t,\ d_k(t)\} \tag{2}$$

is usable for the current task in step t (if $d_k(t) < \Delta t$, the deadline expires within the step).

With maximum local CPU frequency $f_k^{\max}$ (cycles/s), the allocated frequency is $f_k^{\mathrm{loc}}(t) = \alpha_k(t) f_k^{\max}$. The number of bits locally processed within $\tau_k(t)$ is

$$B_k^{\mathrm{loc}}(t) = \min\left\{ Q_k(t),\ \frac{\tau_k(t)\, f_k^{\mathrm{loc}}(t)}{C_k(t)} \right\}, \tag{3}$$

where $C_k(t)$ is the cycles/bit of the current task. The local energy follows a DVFS model:

$$E_k^{\mathrm{loc}}(t) = \varphi\, B_k^{\mathrm{loc}}(t)\, C_k(t)\, \big(f_k^{\mathrm{loc}}(t)\big)^2, \tag{4}$$

where φ is the chip-dependent effective switched capacitance.

After local processing, the remaining bits are $Q_k^{\mathrm{rem}}(t) = Q_k(t) - B_k^{\mathrm{loc}}(t)$. The edge server has total computation capacity f^{CPU} (cycles/s). In our implementation, the CPU allocates resources proportionally to the instantaneous offloaded demand:

$$f_k^{\mathrm{CPU}}(t) = \frac{Q_k^{\mathrm{rem}}(t)}{\sum_{i=1}^{K} Q_i^{\mathrm{rem}}(t) + \epsilon}\, f^{\mathrm{CPU}}, \tag{5}$$

with a small $\epsilon > 0$. Accordingly, the edge service rate (bits/s) is given by

$$r_k^{\mathrm{edge}}(t) = \frac{f_k^{\mathrm{CPU}}(t)}{C_k(t)}. \tag{6}$$

Within the time window $\tau_k(t)$, we adopt a pipelined transmit–compute abstraction: the number of bits that can be both transmitted and executed is approximated by the harmonic-combination throughput

$$S_k^{\mathrm{off}}(t) = \frac{\tau_k(t)}{\frac{1}{R_k(t)+\epsilon} + \frac{1}{r_k^{\mathrm{edge}}(t)+\epsilon}}, \tag{7}$$

and the number of offloaded-and-executed bits is given as

$$B_k^{\mathrm{off}}(t) = \min\{Q_k^{\mathrm{rem}}(t),\ S_k^{\mathrm{off}}(t)\}. \tag{8}$$

The transmission time spent on these bits is $t_k^{\mathrm{tr}}(t) = B_k^{\mathrm{off}}(t)/(R_k(t)+\epsilon)$, and the corresponding offloading energy is

$$E_k^{\mathrm{off}}(t) = p_k(t)\, t_k^{\mathrm{tr}}(t) = \eta_k(t) p_k^{\max}\, t_k^{\mathrm{tr}}(t). \tag{9}$$

Total per-step energy is $E_k(t) = E_k^{\mathrm{loc}}(t) + E_k^{\mathrm{off}}(t)$.

The remaining task bits evolve as

$$Q_k(t+1) = Q_k^{\mathrm{rem}}(t) - B_k^{\mathrm{off}}(t), \tag{10}$$

and the remaining deadline budget updates as $d_k(t+1) = d_k(t) - \Delta t$. A task is *successful* if it completes before the deadline, i.e., $Q_k(t+1) \le 0$ while $d_k(t+1) \ge 0$. If the deadline expires (i.e., $d_k(t+1) \le 0$) and the task is not complete, then only the unprocessed remainder is dropped immediately.

Over an episode of T steps, the goal is to minimize the total energy while meeting application deadlines under heterogeneous task arrivals:

$$\begin{aligned} \min_{\{\alpha_k(t),\, \eta_k(t)\}} \quad & \mathbb{E}\left[\sum_{t=1}^{T} \sum_{k=1}^{K} E_k(t) \right] \\ \text{s.t.} \quad & \tau_k^{(n)} \le D_k^{(n)}, \qquad \forall k,\ \forall n, \\ & 0 \le \alpha_k(t) \le 1, \quad 0 \le \eta_k(t) \le 1, \qquad \forall k,\ \forall t, \end{aligned} \tag{11}$$

The problem in (11) is difficult to solve via conventional convex optimization since the uplink rate is a non-linear SINR function coupled across users, the hard-deadline constraints depend on sequential task evolution, and the system dynamics are driven by stochastic, heterogeneous task arrivals. To obtain an adaptive policy with low online complexity, we adopt a multi-agent deep deterministic policy gradient (MADDPG) [2], which naturally handles continuous actions (DVFS and power control) under decentralized execution, while exploiting centralized training to capture inter-user coupling through shared wireless and edge resources.

III. MADDPG-Based Joint Resource Allocation for Heterogeneous MEC

To address heterogeneous task arrivals, time-varying channels, and hard deadlines, we formulate the proposed joint resource allocation as a cooperative partially observable Markov decision process (POMDP). Each user $k \in \{1, \ldots, K\}$ acts as an agent that selects continuous control variables at every time step t based on local observations, while training is performed using MADDPG under centralized-training decentralized-execution (CTDE) to handle the strong coupling induced by shared wireless and edge resources.

a) Local observation: Since tasks may span multiple time steps, agent k observes the status of its current task and channel quality. We define the local observation as a compact vector $\mathbf{o}_k(t) = \left[\tilde{Q}_k(t),\ \tilde{\kappa}_k(t),\ \delta_k(t),\ \tilde{B}_k(t)\right]$, where $\tilde{Q}_k(t)$ is the normalized remaining bits of the current task, $\tilde{\kappa}_k(t)$ is the normalized cycles/bit, $\delta_k(t) = t_{k,\mathrm{rem}}(t)/(D_k + \epsilon)$ is the remaining-deadline ratio, and $\tilde{B}_k(t)$ is a the total large-scale channel gain from all APs to user k, ($\tilde{B}_k(t) = \sum_{m=1}^{M} \beta_{mk}\,(t)$).

b) Action: At each time step t, user k selects a continuous action $\mathbf{a}_k(t) = \big[\alpha_k(t),\ \eta_k(t)\big], \quad 0 \le \alpha_k(t) \le 1,\ 0 \le \eta_k(t) \le 1$, where $\alpha_k(t)$ controls the local DVFS frequency allocation and $\eta_k(t)$ controls the uplink transmit power for offloading.

c) Reward: The reward is designed to minimize energy while enforcing hard-deadline reliability under heterogeneous arrivals. Let $E_k(t)$ be the per-step energy (local computing plus offloading), and define normalized energy $\tilde{E}_k(t) = \min\{E_k(t)/(E^{\max} + \epsilon), 1\}$. Let the normalized progress be $p_k(t) = \big(Q_k(t) - Q_k(t+1)\big)/(Q_k^{\mathrm{init}} + \epsilon)$ and the remaining fraction be $r_k(t) = Q_k(t+1)/(Q_k^{\mathrm{init}} + \epsilon)$. We use an event-based term to reflect task outcomes and an urgency-shaped

TABLE I
HETEROGENEOUS TASK PROFILES USED IN THE SIMULATION (PER ARRIVAL).

Class c	Task size T (kbits)	Cycles/bit C	Deadline D (ms)
XR	$[20, 80]$	$[300, 800]$	$[10, 20]$
IoT	$[5, 20]$	$[100, 500]$	$[3, 5]$

penalty to discourage carrying large remaining workload close to the deadline:

$$r_k(t) = w_p\, p_k(t) - w_e\, \tilde{E}_k(t) + w_s\, \mathbb{I}_k^{\text{succ}}(t) - w_f\, \mathbb{I}_k^{\text{fail}}(t) - w_u\, \pi_k(t)\, r_k(t)^{\beta}. \tag{12}$$

where $\mathbb{I}_k^{\text{succ}}(t)$ indicates that the current task completes within its deadline, $\mathbb{I}_k^{\text{fail}}(t)$ indicates deadline expiration with unfinished bits, and $\pi_k(t) = 1 - t_{k,\text{rem}}(t)/(D_k + \epsilon)$ increases as the deadline approaches. Finally, we adopt a cooperative global reward shared by all agents, i.e., $r(t) = \frac{1}{K}\sum_{k=1}^{K} r_k(t)$, to encourage coordination to reduce total energy while maintaining deadline compliance across heterogeneous users.

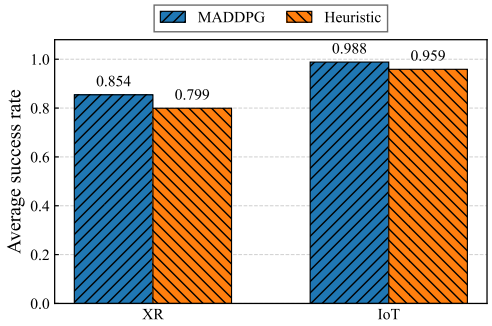
The actor network is a fully-connected feedforward neural network with two hidden layers of 1024 neurons each, using ReLU activations. The final actor head uses a sigmoid squashing function, and the outputs are linearly mapped to satisfy the action bounds. The critic is a centralized fully-connected network that takes the concatenated joint state-action vector of all agents as input, passes it through three hidden layers of 1024 neurons with ReLU activations and outputs a single scalar value estimate. We employ layer normalization in both actor and critic networks for stabilizing the training.

IV. PERFORMANCE EVALUATION

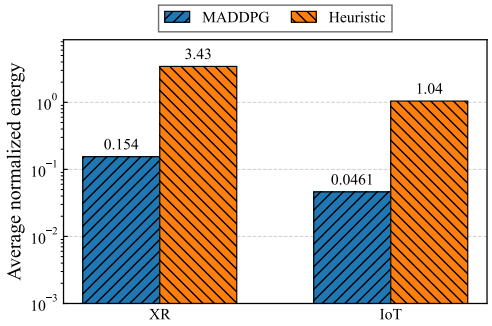
We consider a CF-mMIMO-enabled MEC uplink with $M = 100$ single-antenna APs serving $K = 10$ single-antenna users using MRC combining. The system bandwidth is $W = 20$ MHz, the uplink transmit power is bounded by $P^{\max} = 0.1$ W, and the receiver noise power is computed using a noise figure of 9 dB. Time is slotted with step duration $\Delta t = \tau_s = 1$ ms, and each episode spans $T = 200$ steps (with 400 episodes). The users' maximum local CPU frequency and the edge CPU capacity are set to $f_k^{\max} = 1$ GHz and $f^{\text{CPU}} = 100$ GHz, respectively. Each user belongs to one of two application classes (XR or IoT), drawn with probabilities $p_{\text{XR}} = 0.4$ and $p_{\text{IoT}} = 0.6$; task parameters are sampled per arrival as summarized in Table I (uniformly within the indicated ranges).

As a baseline, we consider a low-complexity heuristic that fully utilizes local computation while allocating uplink transmit power via fractional power control based on the aggregate large-scale gain as $p_k(t) = p_k^{\max} \frac{\left(\sum_{m=1}^{M} \beta_{mk}\right)^{-v}}{\max_{i\in\{1,\ldots,K\}}\left(\sum_{m=1}^{M} \beta_{mi}\right)^{-v}}$, $\forall k$, with $v = 1$, counteracting pathloss disparities.

Fig. 1 compares the MADDPG policy with a heuristic for XR and IoT users. Fig. 1(a) shows the learned policy improves success rates, ensuring reliability. Fig. 1(b) highlights



(a) Average task success rate (deadline satisfaction).



(b) Average normalized energy per task-generation.

Fig. 1. Heterogeneous MEC performance comparison between MADDPG and the heuristic baseline.

the average normalized energy (per-class energy per task-generation opportunity, normalized by the maximum per-user energy per step), showing a significant reduction versus the baseline. Overall, the learned joint control achieves a superior energy-efficiency tradeoff compared to the heuristic.

V. CONCLUSION

We investigated joint resource allocation for CF-mMIMO MEC network with heterogeneous tasks and multi-step deadlines. By modeling class-specific requirements, our learning-based policy adaptively balances energy efficiency and reliability. Results show our approach outperforms heuristic baselines in both reliability and normalized energy across heterogeneous application classes. Future research will explore richer service mixes with dynamic AP clustering and fronthaul constraints, as well as cooperative multi-satellite edge computing.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025-00517140).

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," *IEEE Trans. Veh. Technol.*, vol. 72, no. 12, pp. 16454–16468, Dec. 2023.

[2] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multiagent actor-critic for mixed cooperative-competitive environments," in *Proc. 31st Conf. Neural Inf. Process. Syst.*, Long Beach, CA, 2017, pp. 1–12.