Article:
Du, J., Gong, J., Chu, X. orcid.org/0000-0003-1863-6149 et al. (4 more authors) (2025) QoE-guaranteed optimization in MEC-enabled metaverse: an active inference deep reinforcement learning approach. IEEE Transactions on Cognitive Communications and Networking. ISSN 2332-7731

https://doi.org/10.1109/TCCN.2025.3554003

QoE-Guaranteed Optimization in MEC-Enabled Metaverse: An Active Inference Deep Reinforcement Learning Approach

Jianbo Du, Jie Gong, Xiaoli Chu, Zehui Xiong, Xianfu Chen, Mianxiong Dong, and F. Richard Yu

***Abstract*—In this paper, we consider a MEC-enabled metaverse scenario which consists of a remote metaverse server and an edge server that cooperates to provide services to mobile users. The edge server is deployed at the base station (BS), serves a dual role: augmenting computational capabilities for user equipment (UE) and pre-caching a portion of the metaverse service contents before each time slot. Moreover, the foreground information and the requested contents generated by the UEs can also be cached to the BS. We formulate a problem to maximize the cache hit number by jointly optimizing contents pre-caching and resource allocation at the BS while considering UEs preference and reducing the UEs total energy consumption, essential for the efficient delivery of services in dynamic MEC environments. To solve this problem, we reformulate it as a partially observable markov decision process and propose an active inference enabled deep reinforcement learning algorithm, which combines active inference with deep reinforcement learning to select the optimal strategy by minimizing the expected free energy. Simulations show that the proposed algorithm can effectively improve the total quality of experience and the cache hit number of UEs, while minimizing the UEs total energy consumption compared with other baseline algorithms.**

***Index Terms*—Mobile edge computing, metaverse, deep reinforcement learning, active inference.**

I. Introduction

Metaverse has recently attracted considerable attention from both industry and academia. It enables users to immerse themselves in virtual worlds from any location and at any time, which is a new kind of social ecology that can link physical and virtual worlds [1], offering a multitude of benefits across diverse application scenarios [2]. Nonetheless, the immense demands for data processing pose a significant challenge to the development of the metaverse.

To overcome this challenge, multi-access edge computing (MEC) is a feasible strategy. It situates computational resources in the vicinity of virtual service providers, thereby achieving latencies that are measured in milliseconds. Extensive research has been conducted on MEC in both academic and industrial settings. The authors of [3] constructed a model that includes edge servers and multiple industrial internet of things devices, and proposed an energy-efficient task offloading algorithm to minimize long-term energy consumption. In [4], the authors developed a millimetre wave MEC mechanism based on non-orthogonal multiple access, with the objective of reducing the average delay of MEC task offloading by optimizing beamwidth, user device scheduling and transmission power. The authors of [5] proposed a blockchain-driven joint optimization algorithm for MEC systems to address issues such as high energy consumption during task processing. In the previous paper, we presented a novel space-air-ground integrated network architecture enhanced by MEC and blockchain, which addresses network dynamics by reformulating the original resource allocation problem as a partially observable markov decision process (POMDP) [6].

Due to the low latency and high bandwidth characteristics of the MEC, substantial research have been dedicated to the exploitation of MEC within metaverse applications. Dang et al. proposed an innovative digital twin scheme for metaverse supported by adopting a synergistic model that integrates communication, computation, and storage through MEC and ultra-reliable and low-latency communications [7]. Aung et al. proposed a hybrid architecture of fog-edge for metaverse applications which utilizes edge devices to perform tasks with the heavy computations such as three-dimensional physics calculations within the virtual universe [8]. In [9], the authors introduced an efficient approximation algorithm, which optimizes rewards for augmented reality applications in metaverse-enabled MEC networks, maximizing cumulative rewards within network capacity limits while maintaining application responsiveness. The authors in [10] devised a multiple MEC servers rendering framework, and formulated an optimization problem to maximize the system utility, which contains minimizing energy consumption for MEC servers and

*This work was supported in part by the National Natural Science Foundation of China under Grant 62271391, 62471388, and U24A20209, in part by the Open Research Fund from Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ) under Grant No. GML-KF-24-34, in part by the UKRI EPSRC under Grant EP/X038971/1, in part by Guangdong Province Science and Technology Project 2023A0505050127, in part by Key Research and Development Program of Shaanxi Province under Grant 2024CY2-GJHX-56, and in part by the Serving Local Special Scientific Research Project of Education Department of Shaanxi Province under Grant 21JC032.

Jianbo Du and Jie Gong are with Shaanxi Key Laboratory of Information Communication Network and Security, School of Communications and Information Engineering, Xian University of Posts and Telecommunications, Xian 710121, China. (Email: dujianboo@163.com; gongjie202303@163.com)

Xiaoli Chu is with Department of Electronic and Electrical Engineering, The University of Sheffield, Mappin Street, Sheffield, S1 3JD, UK. (Email: x.chu@sheffield.ac.uk)

Zehui Xiong is with the Information Systems Technology and Design Pillar, Singapore University of Technology and Design, Singapore. (E-mail: zehui_xiong@sutd.edu.sg)

Xianfu Chen is with the Shenzhen CyberAray Network Technology Company Ltd., Shenzhen 518000, China. (E-mail: xianfu.chen@ieee.org)

Mianxiong Dong is with the Department of Sciences and Informatics, Muroran Institute of Technology, Muroran, 050-8585, Japan. (Email: mxdong@csse.muroran-it.ac.jp)

F. Richard Yu is with Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada. (E-mail: Richard.Yu@carleton.ca)

Xiaoli Chu and F. Richard Yu are the corresponding author.

users' quality of experience (QoE). The authors in [11] focused on investigating an UAV-aided vehicular metaverse, paved the way to link metaverse and the real world, and proposed to leverage a proximal policy optimization based deep reinforcement learning algorithm to solve the optimal control policy to the Markov Decision Process (MDP) formulation. In [12], the authors utilized an algorithm that minimizes the average subunit synchronization time between the real world and the digital world, and reduced the average subunit synchronization time between them as much as possible. The integration of MEC in metaverse applications has been a crucial area for researches, leading to innovative solutions that leverage the low latency and high bandwidth of the MEC to enhance user experiences and optimize network operations. However, existing research focus mainly on optimizing the delay or energy consumption of the system to improve the performance, while overlooking the optimization of content caching.

Relying on the advances in MEC for metaverse applications, deep reinforcement learning (DRL) has achieved significant success. The ability of DRL to learn best practices through trial is ideally suited to the dynamic and complex environments common to metaverse applications, enabling intelligent decision making and adaptation to user's behaviour. Mainstream DRL algorithms like Rainbow DQN, proximal policy optimization (PPO) [13], and soft actor-critic (SAC) have been applied to cloud-edge computing environments [14] for task offloading and resource allocation. The authors of [15] established a non-collision control algorithm based on DRL to accomplish waypoint tracking tasks in uncertain circumstances, such as those with unexpected obstacles. In [16], the authors introduced a system based on the DRL algorithm. The authors combined PPO, relative advantage shaping with minimum splitting reward, and deep monte carlo into a self-play framework to facilitate decision-making. In [17], a novel DRL-assisted scheme proposed to co-optimise the task offloading decisions and the allocation of computational resources, etc., based on the content request history and accessible network resources. In [18], the authors proposed a novel framework which combines DRL and lyapunov optimization to solve the coupling problem in different time frames. The authors in [19] propose a federated DT framework to support the imitation of mobile systems. The authors in [20] investigated the secure task offloading and computation resource allocation issues in a consortium blockchain-enabled MEC system, and used PPO to dynamically learn the optimal joint solution to address the problem effectively. The authors in [21] designed an intelligent UAV swarm-based cooperative algorithm for consecutive target tracking and physical collision avoidance, the simulation results demonstrate that the swarm behaviors stay stable in realistic scenarios with perturbing obstacles. The integration of DRL with other optimisation techniques has shown particular efficacy in solving complex challenges such as task offloading and resource allocation, thereby improving the overall efficiency and responsiveness of MEC systems.

Conventional DRL algorithms typically rely on manually designed reward signals to guide agents toward achieving desired objectives. These algorithms predict future states by learning the relationship between past states and optimal actions, optimizing for long-term goals. Due to their focus on long-term returns, short-term dynamics and randomness have minimal impact on their convergence. Moreover, the exploration strategies in DRL help to some extent in avoiding local optimum solution. However, as research advances and the complexity of application scenarios increases, the limitations of conventional DRL algorithms have become increasingly evident. First, the black-box optimization process of DRL makes it challenging to interpret the rationale behind its decision-making, raising concerns about transparency and trustworthiness in critical tasks. Second, mainstream DRL algorithms often encounter convergence issues in complex environments due to the highly dynamic nature of the data and the constraints of single reward designs. In contrast, active inference, grounded in theories from neuroscience such as Bayesian inference and the free energy principle, offers significant advantages. It not only provides strong interpretability but also establishes a natural correspondence between an agent's perception and action mechanisms. Active inference [22], [23] employs an intrinsic generative model that enables agents to actively explore their environment and act based on internal beliefs and preferences. By minimizing the uncertainty between expectations and reality [24], this framework effectively enhances the efficiency of goal-directed behaviors. Compared with traditional DRL algorithms, active inference demonstrates superior adaptability and robustness in complex, dynamic environments, offering a promising solution to the aforementioned challenges.

In recent years, there have been some works applying active inference to the DRL. Fang et al. in [25] developed an innovative algorithm used in cloud-edge network systems for large language models. This approach eschews traditional reward-based guidance in favor of decision-making and resource allocation predicated on the minimisation of expected future free energy. In [26], the authors transformed the trading model into an active inference framework and proposed an intelligence-based reinforcement learning approach that allows for the construction of advanced environmental cognition without the need for external rewards. The above works primarily considered the application of active inference in large language model scenarios and transaction problems. However, research to address the communication issues in MEC-enabled metaverse systems are quite rare.

In summary, while large amount of researches has been dedicated to enhancing system performance by optimizing communication latency, there still lack studies focusing on the full utilization of base station (BS) storage capacity while using active inference methods to further optimize the performance of metaverse systems. The existing literatures indicate that effectively leveraging the storage resources of BS to improve overall system performance remains underexplored. In this work, our main contributions are given as follows.

- We introduce a MEC-enabled metaverse scenario, where a MEC server collaborates with a remote metaverse server that provides metaverse services to multiple user equipments (UEs). The metaverse server caches all metaverse service contents and the MEC server covers a specified metaverse region. We consider the whole communication

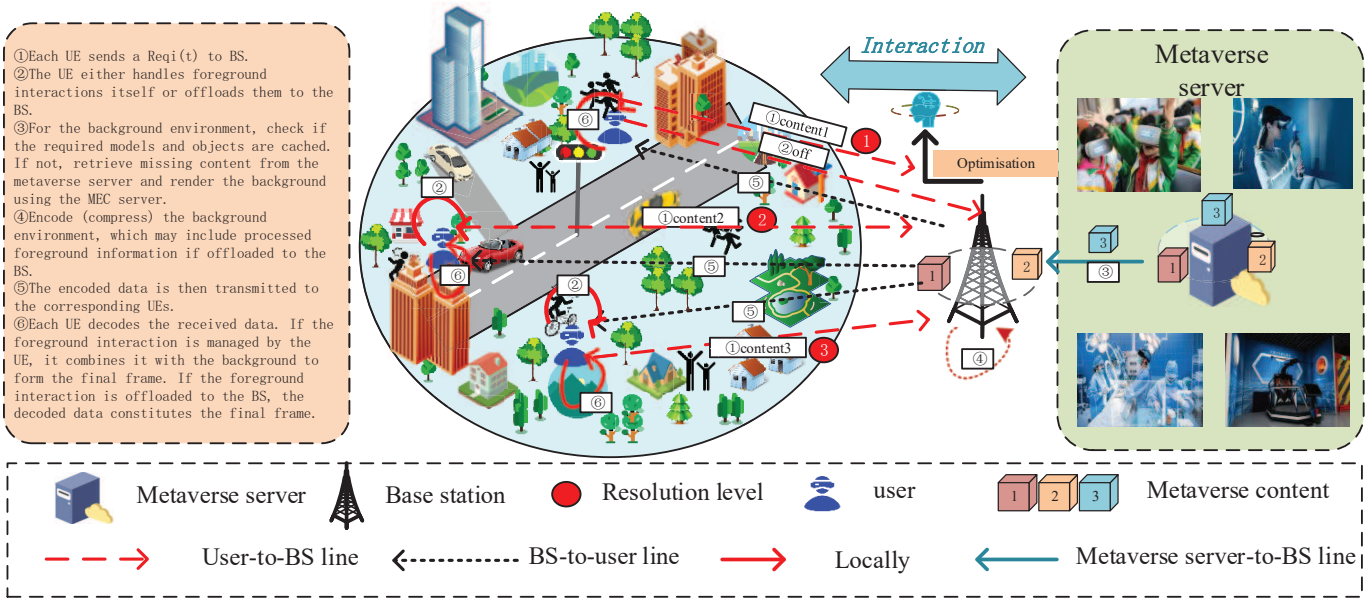

Fig. 1: The concerned scenario.

process, where the UEs first deliver requests to the MEC server, then the MEC server fetches the missed contents from the metaverse server and encodes them before delivering to the UEs. Finally, the UEs decode the contents. In addition, foreground information generated on the user side is also considered, which can be offloaded to the MEC server or processed locally, depending on local processing capabilities.

- Based on the proposed MEC-enabled metaverse model, our objective is to enhance the total QoE and the cache hit number of UEs, while reducing the energy consumption of UEs. The object is achieved through the joint optimization of the cache replacement decision of the background models and objects, the offloading decision of UEs' foreground information, the computation resource allocation of the MEC server, the local computation resource allocation, and the wireless transmission rate allocation of the MEC server. Furthermore, our optimisation is subject to binary constraints for contents caching and replacement, task offloading and the continuous variable constraints on resource allocation, etc.
- Given the randomness of UEs' requests, the highly dynamic nature of channel conditions, and the strong coupling of variables caused by complex constraints, this problem becomes exceptionally intricate and challenging to address. To tackle this, we propose an innovative algorithm-active inference enabled deep reinforcement learning (ADRL). Specifically, ADRL computes the "free energy" through batch simulations of different actions, reducing reliance on existing experiences and predefined objective functions. The algorithm selects the strategies corresponding to several lowest free energy values and parameterizes them as a diagonal Gaussian distribution, with the peak of this distribution determining the strategy selection. By simultaneously maximizing the reward function and minimizing the "expected free energy", the ADRL algorithm achieves faster convergence. Comparative analyses with baseline approaches demonstrate the effectiveness of the proposed algorithm.

The subsequent sections are structured as follows. Section II provides a detailed introduction to the network architecture and system model. In Section III, we describe the problem formulation. In Section IV, the problem is redefined as a MDP problem. Section V elaborates on the proposed ADRL algorithm. In Section VI, we present the simulation results and discussions. Section VII draws the conclusions. Section VIII describes our possible future research directions.

II. Network Architecture

In this section, we consider the uplink and downlink scenarios in a MEC-enabled metaverse network. Specifically, we focus on the metaverse service model, which provides the relative contents of the metaverse. The caching model and offloading model elucidate the mechanisms of content caching and task offloading within the system, respectively. The latency model meticulously delineates the computation of the delay for UEs, a pivotal metric that significantly influences the QoE for UEs. The cache hit model quantifies the frequency of cache hits of the UEs, offering insights into the efficiency of content retrieval. Lastly, the energy analysis model presents a preliminary assessment of the energy consumption of the proposed system, highlighting the trade-offs between performance and energy efficiency. Then, we provide a detailed description of the sub-models.

A. System Description

The MEC enabled metaverse system is illustrated in Fig. 1, and a BS which covers the metaverse region. There are I UEs that request metaverse service, and we consider each UE only make a single request. The set of metaverse UEs is denoted as $\mathcal{I} = \{1, 2, ..., I\}$. The metaverse contents which

TABLE I: Parameter Notations

	Parameter	Notations
Background	Number of Metaverse background models	S
	Number of objects of each model	L_s
	Calculation amount of models	C_s
	Calculation amount of objects	$C_{s,l}$
	Size of models	D_s
	Size of objects	$D_{s,l}$
UEs	Number of Metaverse UEs	I
	Max and min calculation amount of foreground interactions	C^f_{min}, C^f_{max}
	Max and min size of foreground interactions	D^f_{min}, D^f_{max}
	Processing power of UEs	P^p_i
	Transmission power of UEs	P^t_i
	Maximum processing capability of UEs	F^{loc}_{max}
MEC server	Max and min transmission rate between BS and metaverse server	R^b_{min}, R^b_{max}
	Processing capability of the MEC server	F^{mec}
	Maximum transmission rate provided by MEC server	R^{mec}
	Storage capacity of MEC server	Π
Others	Processing density of encoding	λ^{en}
	Processing density of decoding	λ^{de}
	Processing density of frame integration	λ^{inte}
	QoE requirement for end-to-end delay	$\triangle t$
	Weight parameters in objective	$\epsilon_1,\ \epsilon_2,\ \epsilon_3$
	Weight coefficients of UEs' QoE	γ_1
	Weight coefficients of cache hit number	γ_2
	Weight coefficients of UEs' energy consumption	γ_3
	Content life threshold	$Thre$

need to be rendered can be divided into foreground interactions and background virtual environments. In the metaverse, the foreground interaction information encompasses new UE poses and interactions that are triggered by control operations or by interactions with other UEs within the same virtual environment. These interactions tend to be more complex and less predictable, yet they carry a relatively lighter rendering load. Meanwhile, the background virtual environment, which demands a heavier rendering workload, requires a larger storage capacity because of the incorporation of intricate details and complex textures that are essential for the creation of a realistic and immersive experience. The background of each virtual environment is considered to be a model, and there are S models, each model s has some objects. All of these models and objects are cached in a remote metaverse server.

Under specific conditions, when multiple virtual reality users are in the same environment, their visual contents are likely to be very similar, hence they might need to render the same 3D models [27]. In this paper, we assume that the BS can store the rendered contents based on the popularity and the correlation between the different users. Our system operates over discrete in time slots, which is indexed by t, and the set of the time slots is denoted by $\mathcal{T} = \{1, 2, ..., T\}$.

B. Metaverse Service Model

To provide metaverse services, metaverse has two main functionalities, i.e., intensive image processing features that require heavy CPU processing power and background caching associated functionalities [28]. The virtual environment contains two parts, including the background and objects. The set of the backgrounds is denoted as $\mathcal{S} = \{1, 2, ..., S\}$, where each background s is also called model s. Each model contains some objects, which is denoted as $\mathcal{L}_s = \{1, 2, ..., l, ..., L_s\}$. The model s can be described by $\Lambda_s = \{C_s, D_s\}$, $s \in \mathcal{S}$, where C_s (in CPU cycles) is the amount of the computation, i.e., the required CPU cycles to render the model, and D_s (in bits) is the size of the model. Similarly, the object s_l of model s can be described as $\Xi_{s,l} = \{C_{s,l}, D_{s,l}\}$, $s \in \mathcal{S}$, $l \in \mathcal{L}_s$, where $C_{s,l}$ is the required CPU cycles to render the object, and $D_{s,l}$ is the size of the object. The amount of the computation and the size of the models and the objects are constants will keep unchanged. However, UEs may request different model and different objects. The foreground interactions information of a certain UE i in time slot t can be described $\{C^f_i(t), D^f_i(t)\}$, $i \in \mathcal{I}$, $t \in \mathcal{T}$, where $C^f_i(t)$ is the the required CPU cycles, and the $D^f_i(t)$ is the data size, which change across different time slots.

C. Caching Model and Offloading Model

For background, it could be cached on the MEC server and rendered there proactively. Denote the caching indicator of model s as $\kappa_s(t) \in \{0, 1\}$, $s \in \mathcal{S}$, where $\kappa_s(t) = 1$ means model s is cached and rendered on the BS in time slot t, and $\kappa_s(t) = 0$ means that the model need to be downloaded from the metaverse server and rendered by the UE itself locally. Similarly, the caching indicator of object s_l is denoted as $\zeta_{s,l}(t) \in \{0, 1\}$, $s \in \mathcal{S}$, $l \in \mathcal{L}_s$, where $\zeta_{s,l}(t) = 1$ means that object s_l is cached and rendered at the BS in time slot t, and $\zeta_{s,l}(t) = 0$, otherwise.

Denote the storage capacity of the MEC server as Π, when there is no remaining capacity in MEC server, some models and objects (which together called as contents) have to be removed from the MEC server. In this paper, we adopt simple content removal model, i.e., we remove the contents from the MEC server when its life exceeds a certain threshold, thus to keep the freshness and diversity of the cached contents. Define the content life threshold as $Thre$, the life of model s is denoted as $Life_s(t)$, and the life of the object l of model s is defined as $Life_{s,l}(t)$ [29] [30]. The content life is defined as the difference between the indexes of the current time slot and the time slot it is cached into the MEC server, for example, when model s is stored into the model at time slot 3, and the current time slot is time slot 8, the life of model s equals to 8-3=5. Denote the model removal indicator as $v_s(t) \in \{0, 1\}$, if $Life_s(t) > Thre$, the model s will be removed from the MEC server, and we have $v_s(t) = 1$; otherwise, we have $v_s(t) = 0$. The removal indicator of model s can be determined by

$$v_s(t) = \mathbb{I}\{Life_s(t) > Thre\}, \tag{1}$$

where $\mathbb{I}\{\cdot\}$ is the symbolic function, where $\mathbb{I}\{\cdot\} = 1$ when $\cdot$ holds, and $\mathbb{I}\{\cdot\} = 0$ otherwise.

Similarly, we define the object removal indicator as $v_{s,l}(t) \in \{0, 1\}$, and it can be determined by

$$v_{s,l}(t) = \mathbb{I}\{Life_{s,l}(t) > Thre\}. \tag{2}$$

In the above two equations, if a content is not cached at the MEC server, the removal indicator will be zero.

After content caching and removal, we use $n_s(t) \in \{0,1\}$ to denote whether model s exists on the MEC server at time slot t, where $n_s(t)=1$ means it exists, and $n_s(t)=0$ otherwise; meanwhile, we use $n_{s,l}(t) \in \{0,1\}$ to denote whether the object l of model s exists on the MEC server at time slot t, where $n_{s,l}(t)=1$ means it exists, and $n_{s,l}(t)=0$ otherwise.

Based on the above definitions, the content existence indicators evolve according to [31]

$$n_s(t) = n_s(t-1) + \kappa_s(t) \cdot \mathbb{I}\{n_s(t-1)=0\} - v_s(t) \cdot \mathbb{I}\{n_s(t-1)=1\}, \quad (3)$$

$$n_{s,l}(t) = n_{s,l}(t-1) + \zeta_{s,l}(t) \cdot \mathbb{I}\{n_{s,l}(t-1)=0\} - v_{s,l}(t) \cdot \mathbb{I}\{n_{s,l}(t-1)=1\}. \quad (4)$$

Let $\Pi^{rm}(t)$ represents the remaining available capacity of the MEC server, and $\Pi^{rm}(t)$ revolves according to

$$\Pi^{rm}(t) = \Pi^{rm}(t-1) - \sum_{s \notin \mathcal{N}(t-1)} D_s \cdot \kappa_s(t) + \sum_{s \in \mathcal{N}(t-1)} D_s \cdot u_s(t) - \sum_{s \in \mathcal{S}} \sum_{l \notin \mathcal{N}_s(t-1)} D_{s,l} \cdot \zeta_{s,l}(t) + \sum_{s \in \mathcal{S}} \sum_{l \in \mathcal{N}_s(t-1)} D_{s,l} \cdot v_{s,l}(t), \quad (5)$$

where $\mathcal{N}(t-1) = \{s|_{n_s(t-1)=1, s \in \mathcal{S}}\}$ and $N(t-1) = |\mathcal{N}(t-1)|$ are the set and number of models cached in the MEC server at time slot $t-1$; moreover, $\mathcal{N}_s(t-1) = \{l|_{n_{s,l}(t-1)=1, l \in \mathcal{L}_s}\}$ and $N_s(t-1) = |\mathcal{N}_s(t-1)|$ are the set and number of objects of model s cached in the MEC server at time slot $t-1$.

The foreground interactions can be conducted by the UE itself, or be offloaded to the BS. Denote the offloading decision of the foreground interactions of UE i in slot t as $\varrho_i(t)$, $i \in \mathcal{I}$, $t \in \mathcal{T}$, where $\varrho_i(t)=1$ means the foreground interactions are offloaded to the BS, and $\varrho_i(t)=0$ means the foreground interactions are conducted by UE i itself.

D. Latency Model

At the beginning of each time slot, each UE i will generate a metaverse service request, denote the request as $Req_i(t) = \{b_i(t), \mathcal{L}_i(t), C_i^f(t), D_i^f(t), q_i(t)\}$, where $b_i(t) \in \mathcal{S}$ and $b_i(t)=s$ stands for UE i requests to enjoy in background s, $\mathcal{L}_i(t) \subseteq \mathcal{L}_s$ is the set of objects in model s requested by UE i, $C_i^f(t)$ is the CPU cycles needed to render the foreground interactions, $D_i^f(t)$ is the data size of foreground interactions information, and $q_i(t)$ represents the required resolution level of metaverse frames.

For each UE, in order to enjoy metaverse services, there are some typical steps, which are represented in Fig. 1. 1) Each UE sends a request $Req_i(t)$ to the BS. 2) The foreground information can be conducted by the UE itself, or be offloaded to the BS. 3) For the background environments, the BS first examines whether the contents are cached on the BS, where the contents include the requested models and objects. If some of the contents are not cached, the BS first fetches the missed contents from the metaverse server, and then renders them. 4) The BS encodes (or compresses) the background environments, maybe with the processed foreground information if offloaded to the BS. 5) The BS delivers the encoded contents (which may conclude the foreground information if offloaded to the BS) to the corresponding UEs. 6) Each UE decodes the received contents, and if the processing of the foreground information is conducted by the UE, it needs to integrate the foreground and background information into the final frame. However, if the foreground is offloaded to the BS, the decoded contents are the final frame.

1) Foreground Information Rendering Delay: If the foreground information is processed by UE i, the local foreground rendering delay is $T_i^{f,loc}(t) = \frac{C_i^f(t)}{f_i^{loc}(t)}$, where $f_i^{loc}(t)$ is the processing capability of UE i.

If the foreground information is offloaded to the BS, the uplink transmission delay is given by $T_i^{f,up}(t) = \frac{D_i^f(t)}{R_i(t)}$, and the delay that foreground information rendering in the BS is $T_i^{f,rend}(t) = \frac{C_i^f(t)}{f_i^{mec}(t)}$, where $R_i(t)$ is the transmission rate of UE i, and $f_i^{mec}(t)$ is the processing capability that BS allocates to UE i.

2) Background environment Rendering Delay: As mentioned before, background environment information includes model and its embedded objects. Considering $b_i(t)=s$, and the set of requested objects is $\mathcal{L}_i(t)$, the background contents fetching delay from the metaverse server is given in (6), where $R^b(t)$ is the transmission rate between the BS and the remote metaverse server, and the rendering delay is given in (7).

$$t^{b,d}(t) = \frac{1}{R^b(t)} \left(\sum_{s \in \mathcal{S}} \mathbb{I}\{b_i(t)=s\} \cdot \left(\mathbb{I}\{n_s(t-1)=0\} \cdot D_s + \sum_{\{l \in \mathcal{L}_i(t)\} \bigcap \{l \notin \mathcal{N}_s(t-1)\}} D_{s,l} \right) \right), \quad (6)$$

$$t^{b,r}(t) = \frac{\sum_{s \in \mathcal{S}} \mathbb{I}\{b_i(t)=s\} \cdot \left(C_s + \sum_{l \in \mathcal{L}_i(t)} C_{s,l} \right)}{f_i^{mec}(t)}, \quad (7)$$

Therefore, the background rendering delay is

$$T_i^{b,rend}(t) = t^{b,d}(t) + t^{b,r}(t). \quad (8)$$

3) Encode Delay: After rendering, if the foreground information is offloaded to the BS, both the foreground and background information will be encoded. The delay of encoding background is

$$T_i^{b,en}(t) = \frac{\left(D_s + \sum_{l \in \mathcal{L}_i(t)} D_{s,l} \right) \cdot \lambda^{en}}{f_i^{mec}(t)}, \quad (9)$$

where λ^{en} is processing density of encoding.

The delay of encoding foreground information is

$$T_i^{f,en}(t) = \frac{D_i^f(t) \cdot \lambda^{en}}{f_i^{mec}(t)}. \quad (10)$$

4) Encoded Contents Downlink Delivery Delay: If the foreground information is offloaded to the BS, denote the size of encoded background and foreground as $D_i^{b,en}(t) = g\Big(D_s + \sum_{l\in\mathcal{L}_i(t)} D_{s,l},\ q_i(t)\Big)$ and $D_i^{f,en}(t) = g(D_i^f(t),\ q_i(t))$, where $g(x,y)$ is a function of data size and resolution, which is defined as $g(x,y) = \frac{x}{y}$, then the downlink background and foreground encoded contents delivery delay is given by $T_i^{b,down}(t) = \frac{D_i^{b,en}(t)}{R_i(t)}$ and $T_i^{f,down}(t) = \frac{D_i^{f,en}(t)}{R_i(t)}$. Please note that, we consider the wireless channels are reciprocal, where the uplink and downlink data rate is same for each UE.

5) Decode and Obtain Final Frame Delay: If UE i offloads foreground interaction information to the BS, by decoding the received information, the final frame is obtained, and the delay is

$$T_i^{de,off}(t) = \frac{(D_i^{f,en}(t) + D_i^{b,en}(t)) \cdot \lambda^{de}}{f_i^{loc}(t)}. \tag{11}$$

If the foreground interaction information is rendered by UE i locally, the decoding delay is

$$T_i^{de,loc}(t) = \frac{D_i^{b,en}(t) \cdot \lambda^{de}}{f_i^{loc}(t)}. \tag{12}$$

6) Integration Delay: If the foreground interaction information is rendered by UE i locally, it needs to integrate the received information with the local rendered information into the final frame, and the delay is

$$T_i^{inte}(t) = \frac{(D_i^{f}(t) + D_i^{b,en}(t)) \cdot \lambda^{inte}}{f_i^{loc}(t)}. \tag{13}$$

7) Total Latency: If the foreground information is rendered locally, the background processing total delay is

$$T_i^{1,b}(t) = T_i^{b,rend}(t) + T_i^{b,en}(t) + T_i^{b,down}(t) + T_i^{de,loc}(t), \tag{14}$$

and the total delay of UE i is

$$T_i^{tot,1} = \max\Big\{T_i^{f,loc}(t), T_i^{1,b}(t)\Big\} + T_i^{inte}(t). \tag{15}$$

If the foreground information is offloaded to the BS, the total delay is

$$\begin{aligned} T_i^{tot,2}(t) = {} & T_i^{f,up}(t) + T_i^{f,rend}(t) + T_i^{b,rend}(t) \\ & + T_i^{f,en}(t) + T_i^{b,en}(t) + T_i^{f,down}(t) \\ & + T_i^{b,down}(t) + T_i^{de,off}(t). \end{aligned} \tag{16}$$

Combining the two cases, the total delay of UE i is

$$T_i^{tot}(t) = \Big(1 - \varrho_i(t)\Big)T_i^{tot,1}(t) + \varrho_i(t)T_i^{tot,2}(t). \tag{17}$$

Based on the total end-to-end latency, the QoE [32] of UE i is defined as

$$Q_i(t) = \mathbb{I}\{T_i^{tot}(t) \le \Delta t\}, \tag{18}$$

where $\triangle t$ is the maximum tolerable end-to-end delay for enjoying metaverse services.

E. Cache Hit Model

If there is a cache hit for the required model and all required objects, i.e., the required background is all cached in the BS, then the background will be compressed and delivered to UE for further processing and displaying, and we call this situation as a cache hit [28]. Otherwise, if some items of the background are not cached in the BS, the missed items will be fetched from the remote metaverse server and rendered by the MEC server, this will lead to an increased latency [28].

Let $b_i(t) = s$ and $L_i(t) = |\mathcal{L}_i(t)|$, the number of cache hit is considered to be one of our objective, and is defined by

$$z_i(t) = \mathbb{I}\left\{\sum_{s\in\mathcal{S}} \mathbb{I}\{b_i(t) = s\} \cdot n_s(t) \cdot \mathbb{I}\left(\sum_{l\in\mathcal{L}_i(t)} n_{s,l}(t) = L_i(t)\right)\right\}. \tag{19}$$

F. Energy Analysis

The energy consumption of UE i when the foreground information is locally rendered is

$$E_i^1(t) = P_i^p\Big(T_i^{f,loc}(t) + T_i^{de,loc}(t) + T_i^{inte}(t)\Big), \tag{20}$$

where P_i^p is the processing power of UE i. When the foreground information is offloaded to the BS, the energy consumption of UE i is

$$E_i^2(t) = P_i^t T_i^{f,up}(t) + P_i^p T_i^{de,off}(t), \tag{21}$$

where P_i^t is the transmission power of UE i.

Therefore, the energy consumption of UE i is

$$E_i(t) = \Big(1 - \varrho_i(t)\Big)E_i^1(t) + \varrho_i(t)E_i^2(t). \tag{22}$$

III. Problem Formulation

In this section, we first design our objective function, and then formulate the optimization problem.

A. Objective Function

Our objective is to maximize the total QoE of UEs, the cache hit number, and meanwhile to minimize the total energy consumption, and our objective function is designed as

$$Obj(t) = \epsilon_1\gamma_1\sum_{i\in\mathcal{I}} Q_i(t) + \epsilon_2\gamma_2\sum_{i\in\mathcal{I}} z_i(t) - \epsilon_3\gamma_3\sum_{i\in\mathcal{I}} E_i(t), \tag{23}$$

where γ_1 (in Utility), γ_2 (in Utility), γ_3 (in Utility/Joule) are the weight coefficients used to balance the relative magnitude of the QoE, the cache hit number and the energy consumption, ensuring they are on the same scale, and meanwhile to unify the unit of the three items. Moreover, ϵ_1, ϵ_2 and ϵ_3 are used to balance the importance of the three items, which should satisfy $\epsilon_1,\ \epsilon_2,\ \epsilon_3 \in [0,1]$, and $\epsilon_1 + \epsilon_2 + \epsilon_3 = 1$.

B. Problem Formulation

To achieve the objective, we propose to jointly optimize the caching decision of models $\boldsymbol{\kappa}(t) = \{\kappa_s(t)\}$, $s \in \mathcal{S}$, the caching decision of objects $\boldsymbol{\zeta}(t) = \{\zeta_{s,l}(t)\}$, $s \in \mathcal{S}$, $l \in \mathcal{L}_s$, the offloading decision of the foreground interactions of UEs $\boldsymbol{\varrho}(t) = \{\varrho_i(t)\}$, $i \in \mathcal{I}$, the computation resource allocation in the MEC server $\mathbf{f}^{mec}(t) = \{f_i^{mec}(t)\}$, $i \in \mathcal{I}$ and in UEs $\mathbf{f}^{loc}(t) = \{f_i^{loc}(t)\}$, $i \in \mathcal{I}$, and the wireless transmission rate allocation $\mathbf{R}(t) = \{R_i(t)\}$, $i \in \mathcal{I}$, subjecting to the QoE requirement of metaverse UEs, and thus to maximize the objective as defined in Eq. (18). The joint optimization problem is formulated as

$$
\begin{aligned}
(\mathcal{P}_\mathbf{1}): \quad & \max_{\substack{\boldsymbol{\kappa}(t),\boldsymbol{\zeta}(t),\boldsymbol{\varrho}(t) \\ \mathbf{f}^{mec}(t),\mathbf{f}^{loc}(t),\mathbf{R}(t)}} \frac{1}{T}\sum_{t\in\mathcal{T}} Obj(t) \\
\text{s.t. } & (\text{C1}): \kappa_s(t), \zeta_{s,l}(t) \in \{0,1\},\ s \in \mathcal{S},\ l \in \mathcal{L}_s, \\
& (\text{C2}): \varrho_i(t) \in \{0,1\},\ i \in \mathcal{I}, \\
& (\text{C3}): f_i^{mec}(t) \geq 0,\ i \in \mathcal{I}, \\
& (\text{C4}): \sum_{i\in\mathcal{I}} f_i^{mec}(t) \leq F^{mec}, \\
& (\text{C5}): 0 \leq f_i^{loc}(t) \leq f_{max}^{loc},\ i \in \mathcal{I}, \\
& (\text{C6}): R_i(t) \geq 0,\ i \in \mathcal{I}, \\
& (\text{C7}): \sum_{i\in\mathcal{I}} R_i \leq R^{mec}, \\
& (\text{C8}): \sum_{s\in\mathcal{S}} n_s(t) D_s + \sum_{s\in\mathcal{S}}\sum_{l\in\mathcal{L}_s} n_{s,l}(t) D_{s,l} \leq \Pi, \\
& (\text{C9}): \Pi^{rm}(t) \geq 0.
\end{aligned} \tag{24}
$$

In problem $(\mathcal{P}_\mathbf{1})$, (C1) and (C2) require that the background model caching, background objects caching, and foreground interaction information offloading are binary variables. (C3) and (C4) are the constraints on MEC server computation resource allocation, where F^{mec} is the processing capability of the MEC server, and (C4) requires that the total amount of allocated computation resources should not exceed the processing capability of the MEC server. In (C5), f_{max}^{loc} is the maximum processing capability of UEs, which is same for all UEs, and (C5) is the constraint on the value range of local computation resource allocation of UEs. (C6) and (C7) are the constraints on wireless transmission rate allocation, and R^{mec} is the maximum transmission rate that can be provided by the MEC server. (C8) requires that the total size of cached background model and objects should not exceed the storage capacity of the MEC server Π. (C9) requires that the remaining capacity of the MEC server should be non-negative.

IV. Problem Reformulation

Since the high-dynamic and the time-coupled features of the system, it is hard for traditional optimization algorithms to solve the optimal adaptive and real-time decisions. DRL appears as an effective decision making solution, which can mitigate the temporal affects, and focus on long-term objective optimization, and thus is feasible to solve the problems. In the framework of DRL, agent aims to find a policy that can maximize the total expected rewards. Active inference is a promising approach derived from cognitive and computational neuroscience, where agent chooses actions to maximize the evidence supporting a model that is biased towards its own preferences. This framework is an important extension of Bayesian perception and learning to incorporate probabilistic decision making. Although active inference and DRL have different sources, both frameworks use similar approaches to learning adaptive behaviors, and active inference is applied to many established DRL forms, which could provide a promising additional component to current DRL methods, indicate a framework which is potentially unified for adaptive decision making in uncertain circumstances.

A. State Space

According to the description of the system model, the state space is given by $\mathcal{S} = \{s(t), t \in \mathcal{T}\}$, defined as

$$
\begin{aligned}
s(t) \triangleq \Big\{ & \mathbf{C}^f(t),\ \mathbf{D}^f(t),\ \mathbf{b}(t),\ \mathcal{L}(t),\ \mathbf{q}(t),\ R^b(t), \\
& \mathcal{N}(t-1),\ \mathcal{N}_s(t-1),\ \Pi^{rm}(t-1),\ \mathbf{Life}^{md}(t), \\
& \mathbf{Life}^{obj}(t),\ \mathbf{V}^{md}(t),\ \mathbf{V}^{obj}(t) \Big\},
\end{aligned} \tag{25}
$$

where

- $\mathbf{C}^f(t) = \{C_i^f(t)\}$, $i \in \mathcal{I}$, where $C_i(t)$ is the CPU cycles needed to render the foreground interactions of UE i in time slot t;
- $\mathbf{D}^f(t) = \{D_i^f(t)\}$, $i \in \mathcal{I}$, where $D_i(t)$ is the data size of foreground interaction information of UE i in time slot t;
- $\mathbf{b}(t) = \{b_i(t)\}$, $i \in \mathcal{I}$, where $b_i(t) \in \mathcal{S}$ is the requested index of background model of UE i in time slot t;
- $\mathcal{L}(t) = \{\mathcal{L}_i(t)\}$, $i \in \mathcal{I}$, where $\mathcal{L}_i(t) \in \mathcal{L}_s$ is the requested set of background objects of UE i in time slot t;
- $\mathbf{q}(t) = \{q_i(t)\}$, $i \in \mathcal{I}$, where $q_i(t)$ is the required resolution level of metaverse frames of UE i in time slot t;
- $R^b(t)$ is the wireless transmission rate between the BS and the remote metaverse server in time slot t;
- $\mathcal{N}(t-1) = \{s|_{n_s(t-1)=1, s\in\mathcal{S}}\}$ is the set of models cached in the MEC server at time slot $t-1$;
- $\mathcal{N}_s(t-1) = \{l|_{n_{s,l}(t-1)=1, l\in\mathcal{L}_s}\}$ is the set of objects of model s cached in the MEC server at time slot $t-1$;
- $\Pi^{rm}(t-1)$ is the remaining available caching resource of the MEC server after content placement at time slot $t-1$;
- $\mathbf{Life}^{md}(t) = Life_s(t), s \in \mathcal{S}$ is the life of model s at time slot t;
- $\mathbf{Life}^{obj}(t) = Life_{s,l}(t), s \in \mathcal{S}, l \in \mathcal{L}$ is the life of object l of model s at time slot t;
- $\mathbf{V}^{md}(t) = \{v_s(t)\}, s \in \mathcal{S}$ is the removal indicator of model s at time slot t;
- $\mathbf{V}^{obj}(t) = \{v_{s,l}(t)\}, s \in \mathcal{S}, l \in \mathcal{L}$ is the removal indicator of object l of model s at time slot t.

B. Action Space

In each time slot, the agent decides its actions $\mathbf{a}(t)$, which comprises the following items.

- The caching decision of models $\boldsymbol{\kappa}(t) = \{\kappa_s(t)\}$, $s \in \mathcal{S}$;

- The caching decision of objects $\boldsymbol{\zeta}(t) = \{\zeta_{s,l}(t)\}$, $s \in \mathcal{S}$, $l \in \mathcal{L}_s$;
- The offloading decision of the foreground interactions of UEs $\boldsymbol{\varrho}(t) = \{\varrho_i(t)\}$, $i \in \mathcal{I}$;
- The computation resource allocation in the MEC server $\mathbf{f}^{mec}(t) = \{f_i^{mec}(t)\}$, $i \in \mathcal{I}$;
- The computation resource allocation in UEs $\mathbf{f}^{loc}(t) = \{f_i^{loc}(t)\}$, $i \in \mathcal{I}$;
- The wireless transmission rate allocation $\mathbf{R}(t) = \{R_i(t)\}$, $i \in \mathcal{I}$.

Thus, the action in time slot t is given by

$$\mathbf{a}(t) \triangleq \{\boldsymbol{\kappa}(t),\ \boldsymbol{\zeta}(t),\ \boldsymbol{\varrho}(t),\ \mathbf{f}^{mec}(t),\ \mathbf{f}^{loc}(t),\ \mathbf{R}(t)\}. \quad (26)$$

C. Reward Function

In problem $(\mathcal{P}_\mathbf{1})$, our objective is to maximize the long-term average objective of the system. Since the immediate reward reflects the weighted sum of QoE of UEs, cache hit number, and the energy consumption of UEs at each time slot, we consider the immediate reward $r(t)$ as

$$r^{imm}(t) = \epsilon_1\gamma_1 \sum_{i\in\mathcal{I}} Q_i(t) + \epsilon_2\gamma_2 \sum_{i\in\mathcal{I}} z_i(t) - \epsilon_3\gamma_3 \sum_{i\in\mathcal{I}} E_i(t). \quad (27)$$

Considering the constraints in $(\mathcal{P}_\mathbf{1})$, if the constraints are all satisfied, the agent will be rewarded with $r^{imm}(t)$, otherwise, the agent will not obtain any reward, since the solution it finds is not feasible. We use a binary variable $v(t)$ as the indicator, where if all constraints are satisfied, we have $v(t) = 1$, and the immediate reward is set to the system utility in the current time slot t; otherwise, if the action violates any of the constraints, the system incurs a penalty. In this paper, we set $v(t) = 0$, and therefore, $r(t) = 0$ in such cases. Accordingly, the immediate reward $r(t)$ is revised as

$$r(t) = v(t)r^{imm}(t). \quad (28)$$

D. Next State

After action selection and execution, the agent will receive an immediate reward. The system state transitions from the current state $\mathbf{s}(t)$ to the next state $\mathbf{s}(t+1)$ according to the following dynamics:

- The CPU cycles while render the foreground interactions of the UE i in time slot $t+1$, $\mathbf{C}^f(t+1) = \{C_i^f(t+1)\}$, $i \in \mathcal{I}$, is generated from $[C_{min}^f, C_{max}^f]$ randomly;
- The data size of foreground information of the UE i in time slot $t+1$, i.e., $\mathbf{D}^f(t+1) = \{D_i^f(t+1)\}$, $i \in \mathcal{I}$, is generated from $[D_{min}^f, D_{max}^f]$ randomly;
- The requested index of background model of the UE i in time slot $t+1$, i.e., $\mathbf{b}(t+1) = \{b_i(t+1)\}$, $i \in \mathcal{I}$, $b_i(t+1) \in \mathcal{S}$, is chosen from $\{1, 2, ..., S\}$ randomly;
- The requested set of background objects of the UE i in time slot $t+1$, i.e., $\mathcal{L}(t+1) = \{\mathcal{L}_i(t+1)\}$, $i \in \mathcal{I}$, $\mathcal{L}_i(t+1) \subseteq \mathcal{L}_s$, is generated like this, first generate a random value l from $\{1, 2, ..., |\mathcal{L}_i(t+1)|\}$, and then randomly choose l objects from $\mathcal{L}_s$;
- The required resolution level of mateverse frames of the UE i in time slot $t+1$, i.e., $\mathbf{q}(t+1) = \{q_i(t+1)\}$, $i \in \mathcal{I}$, is chosen from $\{1, 2, 3, 4\}$ randomly;
- The wireless transmission rate between the BS and the remote metaverse server in time slot $t+1$, i.e., $R^b(t+1)$ is generated from $[R_{min}^b, R_{max}^b]$ randomly;
- The set of models cached in the MEC server at time slot t, i.e., $\mathcal{N}(t) = \{s|_{n_s(t)=1, s\in\mathcal{S}}\}$, updates according to Eq. (3);
- The set of objects of model s cached in the MEC server at time slot t, i.e., $\mathcal{N}_s(t) = \{l|_{n_{s,l}(t)=1, l\in\mathcal{L}_s}\}$, evolves according to Eq. (4);
- The remaining available caching resource of the MEC server after content placement at time slot t, i.e., $\Pi^{rm}(t)$, evolves according to Eq. (5);
- The life of model s at time slot $t+1$, i.e., $\mathbf{Life}^{md}(t+1) = Life_s(t+1), s \in \mathcal{S}$ is updated according to $Life_s(t+1) = Life_s(t) + 1$;
- The life of object l of model s at time slot $t+1$, i.e., $\mathbf{Life}^{obj}(t+1) = Life_{s,l}(t+1), s \in \mathcal{S}, l \in \mathcal{L}$ is updated according to $Life_{s,l}(t+1) = Life_{s,l}(t) + 1$;
- The removal indicator of model s at time slot t, i.e., $\mathbf{V}^{md}(t) = \{v_s(t)\}, s \in \mathcal{S}$, is determined according to Eq. (1);
- The removal indicator of object l of model s at time slot $t+1$, i.e., $\mathbf{V}^{obj}(t+1) = \{v_{s,l}(t+1)\}, s \in \mathcal{S}, l \in \mathcal{L}$ is determined according to Eq. (2).

Remark 2: In the following, we use s_t, a_t and s_{t+1} to represent $s(t)$, $a(t)$ and $s(t+1)$ for notation simplicity.

V. Active Inference Enabled DRL Based Joint Optimization Algorithm

In this section, we will first present the preliminaries of ADRL algorithm, and then present the pseudocode of the ADRL algorithm.

A. Preliminaries of ADRL

ADRL is a DRL algorithm based on active inference, which is well-suited for addressing problems framed as partially observable Markov decision processes (POMDPs). POMDPs involve a set of states, actions, transition probabilities, rewards, observations, and conditional probabilities for outcomes. In this framework, agents inferring the hidden state from partial observations complicates decision-making.

Active inference, initially designed for POMDPs, can also enhance fully observable MDP by incorporating intrinsic motivation, thus improving decision-making efficiency. By combining active inference with DRL, we can optimize both reward and free energy, leading to faster convergence and better policy selection even in environments with low uncertainty.

The primary goal of both active inference and DRL is to enable intelligent agents to make optimal decisions in uncertain environments. However, traditional DRL faces challenges in generalizing policies and adapting to dynamic settings, as it relies heavily on value functions and exploration strategies, typically requiring many trials.

In contrast, active inference operates without explicitly defined reward functions; it uses an internal generative model to drive actions based on intrinsic beliefs, aiming to reduce uncertainty. This framework is grounded in the free energy principle, which suggests that systems maintain equilibrium by minimizing discrepancies between expected beliefs and actual observations. By minimizing free energy, agents can achieve adaptable, goal-directed behaviors, making active inference a powerful approach for understanding perception, learning, and decision-making in both biological systems and machine learning.

The conceptual process of strategy selection depends on the expected future free energy of the agent. Variational free energy (VFE) constitutes a tractable bound on the KL divergence between the logarithm of model evidence and prior and posterior, defined as

$$F = KL\big(\delta(s,\theta)\big\|\eta^{\Phi}(m,s,\theta)\big), \tag{29}$$

where $\delta(s,\theta)$ is the agent's belief about future variables, and $\eta^{\Phi}(m,s,\theta)$ is the generative model with preferences. By setting the preferences, $\eta^{\Phi}(m,s,\theta)$ can be made as close as possible to $\eta^{\Phi}(m,s,\theta)$, that is, to minimize the variational free energy for each time slot. Therefore, the VFE is also referred to as the lower bound of (negative) model evidence.

Since the F solved at each time slot, but the optimization policy π is often a time series, therefore, we expand the F to include future variables, thereby obtaining the expected future free energy $\tilde{F}$ (the KL divergence between the agent's beliefs about future variables and the agent's preferred generative model). The current goal is to minimize $\tilde{F}$ to obtain the optimal policy π^*, which is defined by

$$\tilde{F} = KL(\delta(m_{0:T}, s_{0:T}, \theta, \pi) \parallel \eta(m_{0:T}, s_{0:T}, \theta)), \tag{30}$$

where $m_{0:T}$ represents the agent's observation sequence on the time series $0:T$, $s_{0:T}$ represents the agent's state sequence on the time series $0:T$, $\delta(m_{0:T}, s_{0:T}, \theta, \pi)$ represents the agent's beliefs about future variables, $\eta(m_{0:T}, s_{0:T}, \theta)$ is the agent's generative model, θ is the parameters neural network of the generative model. It is known from the properties of divergence that it is non-negative, so the minimum value of $\tilde{F}$ is 0 only when $\eta(m_{0:T}, s_{0:T}, \theta)$ equal to $\delta(m_{0:T}, s_{0:T}, \theta, \pi)$, that is

$$KL(\delta(m_{0:T}, s_{0:T}, \theta, \pi)\|\eta(m_{0:T}, s_{0:T}, \theta)) = 0 \Longrightarrow \tilde{F} = 0. \tag{31}$$

In order to obtain a optimal policy $\delta(\pi)$ by minimizing the $\tilde{F}$, we mention that [33]

$$\tilde{F} = 0 \Rightarrow KL\left(\delta(\pi) \parallel e^{-\tilde{F}_{\pi}}\right) = 0, \tag{32}$$

where

$$\tilde{F}_{\pi} = KL\left(\delta\big(m_{0:T}, s_{0:T}, \theta|\pi\big)\|\eta^{\Phi}(m_{0:T}, s_{0:T}, \theta)\big)\right), \tag{33}$$

when $\delta(\pi) = \sigma(-\tilde{F}_{\pi})$, the expected future free energy is minimized, or in other words, when the $\tilde{F}_{\pi}$ is minimized, the strategy is more likely to occur.

According to [33], $-\tilde{F}_{\pi}$ can be split into expected information gain terms and external terms, that is

$$-\tilde{F}_{\pi} \approx \underbrace{\mathbb{E}_{m_{0:T}|\pi)}\left[KL\left(\delta(s_{0:T}, \theta \mid m_{0:T}, \pi) \parallel \delta(s_{0:T}, \theta \mid \pi)\right)\right]}_{\text{Expected information Gain}} - \underbrace{\mathbb{E}_{\delta(s_{0:T},\theta|\pi)}\left[KL\left(\delta(m_{0:T} \mid s_{0:T}, \theta, \pi) \parallel \eta^{\Phi}(m_{0:T})\right)\right]}_{\text{Extrinsic Term}}, \tag{34}$$

where the first term maximizes the expected information gain, promoting exploration of the state space. In addition, the second term minimizes the KL difference between the agent's beliefs about future observations and its preferred observations. Overall, minimizing $\tilde{F}_{\pi}$ achieves a harmonious equilibrium between exploration and development.

In each time slot, we need to optimize to select the most likely action. This requires the calculation of three parts: 1) the calculation of beliefs about future variables; 2) the calculation of $\tilde{F}_{\pi}$; 3) the optimization of $\delta(\pi)$.

1) Calculation of beliefs about future variables: Beliefs about future variables can be decomposed as

$$\delta(s_{0:T}, m_{0:T}, \theta|\pi) = \delta(\theta)\prod_{t=0}^{T}\delta(m_t|s_t,\theta,\pi)\delta(s_t|s_{t-1},\theta,\pi), \tag{35}$$

where the observation m_t is correspond directly to the current state s_t in fully observable MDP.

$$\delta(m_t|s_t,\theta,\pi) = \mathbb{E}_{\delta(s_t|\theta,\pi)}[\eta(m_t|s_t)], \tag{36}$$

$$\delta(s_t|s_{t-1},\theta,\pi) = \mathbb{E}_{\delta(s_{t-1}|\theta,\pi)}[\eta(s_t|s_{t-1},\theta,\pi)]. \tag{37}$$

2) Calculation of $\tilde{F}_{\pi}$: The $\tilde{F}_{\pi}$ for each time slot is

$$\begin{aligned} -\tilde{F}_{\pi_\tau} &\approx -\mathbb{E}_{\delta(s_t,\theta|\pi)}\left[KL\Big(\delta(m_t|s_t,\theta,\pi)\Big\|\eta^{\Phi}(m_t)\Big)\right] \\ &- \mathbf{H}[\delta(m_t \mid \pi)] + \mathbb{E}_{\delta(\mathbf{s}_t|\pi)}[\mathbf{H}[\delta(m_t \mid s_t, \pi)]] \\ &- \mathbf{H}[\delta(s_t \mid s_{t-1}, \theta, \pi)] + \mathbb{E}_{\delta(\theta)}\Big[\mathbf{H}[\delta(s_t \mid s_{t-1}, \theta, \pi)]\Big]. \end{aligned} \tag{38}$$

Here, since the state is determined and has no uncertainty, the value of the second term in the above formula is 0. Furthermore, the first term can be calculated by [34]

$$-\mathbb{U}\left(o_\tau\right) \propto \mathbb{E}_{\delta(s_t,\theta|\pi)}\left[KL\Big(\delta(m_t|s_t,\theta,\pi)\Big\|\eta^{\Phi}(m_t)\Big)\right], \tag{39}$$

where $\mathbb{U}\left(o_\tau\right)$ represents the cumulative reward of the current step, i.e.,

$$\mathbb{U}\left(o_\tau\right) = \sum_{t=0}^{\tau} r(t), \tag{40}$$

where $r(t)$ is defined in Eq. (28) by the (P1).

Therefore, for the proposed MDP model, Eq. (38) is transformed as

$$\begin{aligned} -\tilde{F}_{\pi_\tau} &\approx \sum_{t=0}^{\tau} r(t) - \mathbf{H}[\delta(m_t \mid \pi)] + \mathbb{E}_{\delta(\mathbf{s}_t|\pi)}[\mathbf{H}[\delta(m_t \mid s_t, \pi)]] \\ &- \mathbf{H}[\delta(s_t \mid s_{t-1}, \theta, \pi)] + \mathbb{E}_{\delta(\theta)}\Big[\mathbf{H}[\delta(s_t \mid s_{t-1}, \theta, \pi)]\Big]. \end{aligned} \tag{41}$$

3) Optimization of $\delta(\pi)$: We parameterize $\delta(\pi)$ as a diagonal Gaussian distribution, so only the peak of $\tilde{F}_{\pi}$ needs to be obtained to determine the policy selection.

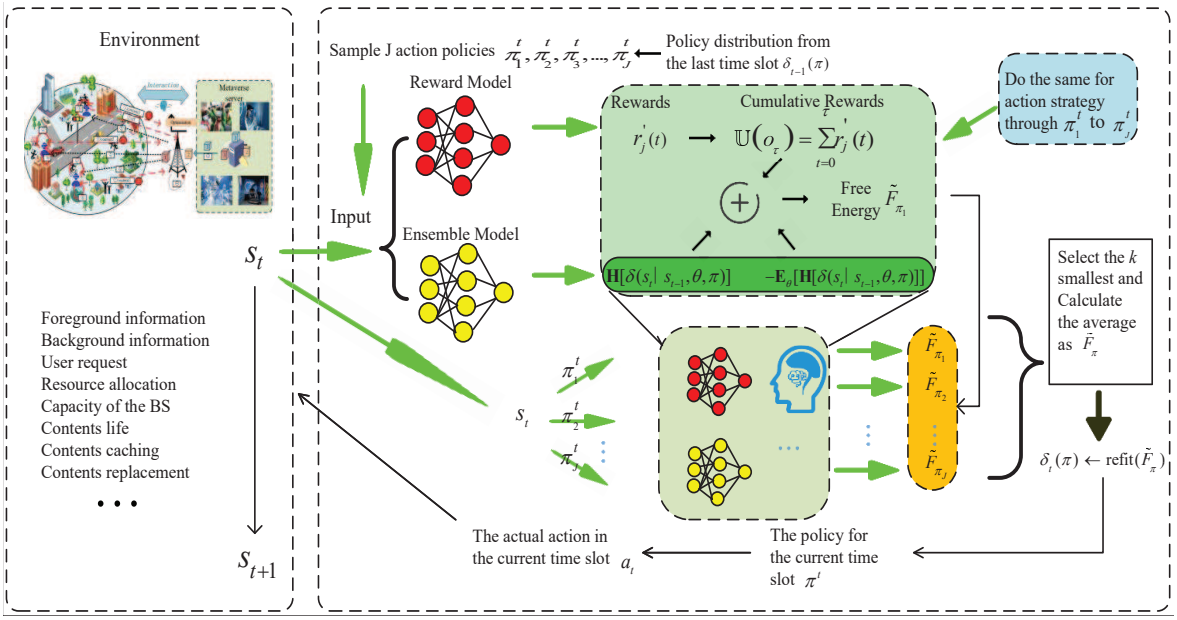

Fig. 2: Flowchart of our proposed Algorithm 1.

B. Pseudocode of ADRL Algorithm

The detailed pseudocode for the ADRL-based joint content caching, offloading decision, base station computational resource allocation, and base station transmission rate allocation algorithm can be found in Algorithm 1. Since ADRL is designed to address problems with continuous action spaces, the agent will output continuous actions. To adapt the outputs to our defined problem (P1), we need to transform the outputs.

1) Continuous Action Normalization: To reduce the dimensionality of the action space, all continuous variables (such as computation resource allocation) are constrained to a value between 0 and 1. Let a represent a general action output, with a^{max} and a^{min} denoting the maximum and minimum values that a can take, respectively, and let $\bar{a}$ denote the actual value. Then, $\bar{a}$ can be recovered from a as follows:

$$\bar{a} = \frac{a - a^{\min}}{a^{\max} - a^{\min}}. \tag{42}$$

2) Discrete Action Reformulation: We transform the discrete actions as follows: For the continuous actions, the output of the proposed algorithm keeps unchanged; for the binary variables, we consider the continuous outputs as the probabilities of $\kappa_s(t) = 1$, $\zeta_{s,l}(t) = 1$ and $\varrho_i(t) = 1$, respectively, during the training phase, and the values of them are obtained by sampling with the probabilities. During the testing phase, the values of $\kappa_s(t)$, $\zeta_{s,l}(t)$ and $\varrho_i(t)$ are obtained by rounding the output continuous values to the nearest integer.

The flowchart of our proposed Algorithm 1 is illustrated in Fig. 2 in order to facilitate a better understanding of the working principle.

VI. Simulation Results and Discussions

In this section, we carry out simulation experiments to examine the performance of our ADRL algorithm and discuss the experimental results. The simulation experiments were conducted utilizing a laptop. The implementation was carried

Algorithm 1 ADRL-based Joint Optimization Algorithm

Initialization:

Initialize: initial state s_t, ensemble network θ_1, reward network θ_2, factorized belief overaction sequences $\delta(\pi) \leftarrow \mathcal{N}(0, \Pi)$, Parameter distribution $\eta(\theta)$;
Input: planning horizon H, optimization iterations N, number of candidate policies J, the number of episodes T_{max}, the number of train epochs N_{ep};

Iteration:

while episode $T \leq T_{max}$ **do**
  Reset the environment;
  while epoch $e \leq N_{ep}$ **do**
    Get $batch_{size}(s_t, a_t, r_t, s_{t+1})$ from buffer;
    Perform a gradient desent step on the ensemble network parameters θ_1;
    Perform a gradient desent step on the reward network parameters θ_2;
  end while
  Get s_t by interact with the environment;
  Initial the $\delta(\pi)$ by $\mathcal{N}(0, 1)$;
  while optimisation iteration $n \leq N$ **do**
    Sample J candidate policies from $\delta(\pi)$;
    while candidate policy $j \leq J$ **do**
      Get the $\mathbf{H}[\delta(s_t \mid s_{t-1}, \theta, \pi)]$ and the $\mathbb{E}_{\delta(\theta)}\Big[\mathbf{H}[\delta(s_t \mid s_{t-1}, \theta, \pi)]\Big]$ according to ensemble network θ_1;
      Get predicted reward $r_t^{'}$ according to reward network θ_2;
      Update $-\tilde{F}_{\pi}^{j}$ by Eq.(41);
    end while
    Select top $k(k \leq j)$ π_j according to $-\tilde{F}_{\pi}^{j}$;
  end while
  Optimize π according to top $k\pi_j$;
  $\delta(\pi) \leftarrow \text{refit}(\tilde{F}_{\pi}^{j})$;
  Take action $a_t \sim \pi$;
  Reform action a_t as $\bar{a}_t$;
  Get s_{t+1}, r_t by step$(\bar{a}_t)$;
  Let $s_{t+1} = s_t$;
end while
return $\pi^* = \pi$.

out using python 3.8, in conjunction with TensorFlow 2.3, providing a robust environment for executing the computational tasks associated with our research. The default parameters are listed in Table II, and the default learning rate is set at 0.01.

TABLE II: Simulation Parameters

Notations	Value
I	10
S	5
L_s	{6,7,8,9} randomly
C_s	[200,600] M CPU cycles randomly
$C_{s,l}$	[20,100] M CPU cycles randomly
D_s	[20,60] Mbit randomly
$D_{s,l}$	[2,30] Mbit randomly
C^f_{min}, C^f_{max}	{50,200} M CPU cycles [27]
D^f_{min}, D^f_{max}	{10,40} Mbit
R^b_{min}, R^b_{max}	{10,500} Mbps
λ^{en}	5 CPU cycles/bit
λ^{de}	5 CPU cycles/bit
λ^{inte}	5 CPU cycles/bit
$\triangle t$	25 ms [35]
P^p_i	10W $\sim$ 20W randomly
P^t_i	0.1W $\sim$ 2W randomly [36]
$\epsilon_1,\ \epsilon_2,\ \epsilon_3$	$\frac{1}{3},\ \frac{1}{3},\ \frac{1}{3}$
γ_1	0.1 Utility
γ_2	10 Utility
γ_3	10 Utility/Joule
F^{mec}	10 G CPU cycles/s
F^{loc}_{max}	1 G CPU cycles/s [37]
R^{mec}	1 Gbps [38]
Π	1.7Gbit
$Thre$	5

In the following simulations, we compare the proposed algorithm with the following benchmark algorithms.

1) No-model-caching: The caching indicator of all models, (i.e., $\kappa_s(t)$) are all set for 0 at the start of each time slot, i.e., the models that requested by UEs are not cached at the BS, and need to download them from the remote metaverse server. This benchmark algorithm is employed to evaluate the impact of the model cache decision in the proposed model framework.

2) No-object-caching: The caching indicator of all objects (i.e., $\zeta_{s,l}(t)$) are all set for 0.

3) Average-MEC-computation: The processing capability of the BS (i.e., F^{mec}) is average-allocated for each UE i. This means that each UE can fairly obtain a certain proportion of computing resources. This method of allocation helps to ensure that all UEs can receive a similar level of service performance.

4) Average-transmission-rate: This algorithm evenly allocates the transmission rate provided by the BS to each UE. This approach ensures that every UE in the network can enjoy a balanced data transmission speed.

5) DDPG: This is a traditional DRL algorithm which is developed for decision making under continuous action space. Since our formulated problem involves both continuous and integer variables within the action space, in order to used DDPG to solve our problem, it is tailored like this: for the continuous actions, the output of DDPG keeps unchanged; for the binary variables, we consider the continuous outputs as the probabilities of $\kappa_s(t) = 1$, $\zeta_{s,l}(t) = 1$ and $\varrho_i(t) = 1$, respectively, during the training phase, and the values of them are obtained by sampling with the probabilities. During the testing phase, the values of $\kappa_s(t)$, $\zeta_{s,l}(t)$ and $\varrho_i(t)$ are obtained by rounding the output continuous values to the nearest integer.

Algorithms typically require metrics to characterize their performance and effectiveness [39]. Based on a comprehensive evaluation from various perspectives, we conduct a comparative analysis of the aforementioned algorithms using the following indicators: system service quality, cache accuracy, overhead, and reward.

1) System service quality can be reflected through the QoE of the UEs. When the delay in content retrieval decreases, the real-time performance of the system improves, resulting in an enhanced user experience; conversely, an increase in delay leads to a poorer user experience. The framework we propose dynamically optimizes cached content and adjusts resource allocation to minimize the total delay in content retrieval, thereby improving system service quality.

2) Cache accuracy is measured by the cache hit number. The higher the proportion of content pre-cached at the base station that matches user requests during each time slot, the higher the cache hit rate. This leads to shorter transmission paths for users acquiring content, thereby reducing transmission delays, which in turn enhances the real-time performance of the system.

3) Overhead primarily includes various costs incurred at the user end, with user energy consumption being a significant component. The energy consumption at the user end mainly arises from uploading front-end information, decompressing received content, and merging processes. By optimizing the local sending power, processing power, and processing capability of users, it is possible to effectively reduce energy consumption at the user end and consequently save on user overhead.

4) Reward is composed of a weighted sum of QoE, cache hit number, and energy consumption, as defined in Eqs. (27) and (28), and the three weight factors $\epsilon_1,\ \epsilon_2,\ \epsilon_3$ take their default values, i.e., the value of each of them is set to $\frac{1}{3}$, and thus the three components contribute equally in the metrics.

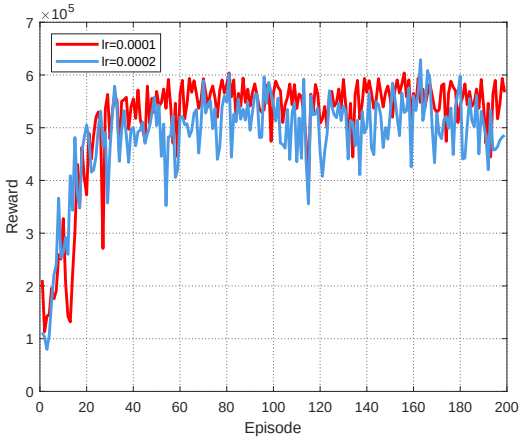

Fig. 3: Convergence under different learning rates of the ADRL agent.

A. Convergence

In this section, we evaluate the convergence behavior of our proposed algorithm across various learning rates.

Fig. 3 shows the convergent trend of the proposed algorithm when the learning rate is 0.0001 and 0.0002 respectively. As shown in this figure, the rewards increase fast in the first 40 rounds, and then gradually converges around the 60th episode under different learning rates, which demonstrate that our proposed algorithm can converge fast.

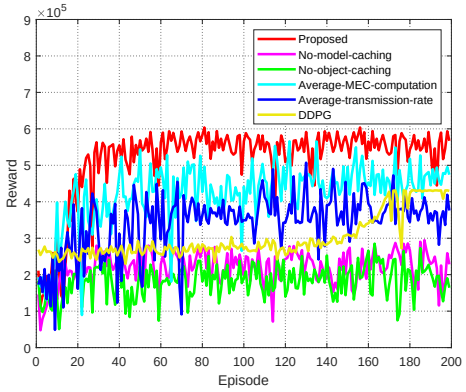

Fig. 4: Comprehensive convergence performance comparison.

Fig. 4 illustrates the convergence performance of the proposed algorithm compared to the benchmark algorithms. The convergence performance can be evaluated in terms of convergence speed and total reward, etc. It is evident that most algorithms achieve convergence around episode 60, while the DDPG algorithm completes convergence around episode 180, indicating that the DDPG algorithm requires more exploration and adjustment. This phenomenon is primarily attributed to the dual driving mechanism of free energy and reward adopted by the proposed algorithm, integrating neural network optimization with biological characteristics. Compared to traditional DRL algorithms that rely solely on a single reward signal, the proposed algorithm considers additional factors affecting the agent's behavior during the decision-making process. Furthermore, after convergence, the proposed algorithm can achieve the highest reward value, while the reward values of other algorithms are relatively lower.

B. Performance Evaluation

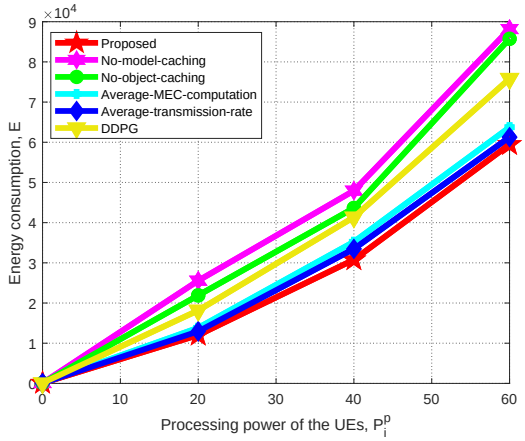

Fig. 5: Energy consumption, E vs. the processing power of the UEs, P_i^p.

In Fig. 5, we demonstrate how the energy consumption of the UEs changes as its processing capability (i.e., P_i^p) increases. It can be observed that when P_i^p = 0.001W, the energy consumption of the UEs is also relatively low. There is a positive correlation between them, which can be reflected by Eqs. (20) and (21). As P_i^p further increases, the energy consumption of the UEs also rises. It is evident from Fig. 5 that compared with the proposed algorithm, the No-model-caching, No-object-caching, and DDPG algorithms have higher energy consumption, while the performance of the Average-MEC-computation and Average-transmission-rate algorithms is similar to that of our algorithm. Since when the processing capacity changes, the allocation of processing capacity and the maximum transmission rate of the MEC server primarily affect the BS's processing delay and transmission delay, while having a relatively small impact on the delays associated with energy consumption. In addition, the figure also shows that the proposed algorithm surpasses other benchmark algorithms (including the traditional DRL algorithm DDPG) in terms of minimizing system energy consumption across systems with varying UEs' processing capabilities.

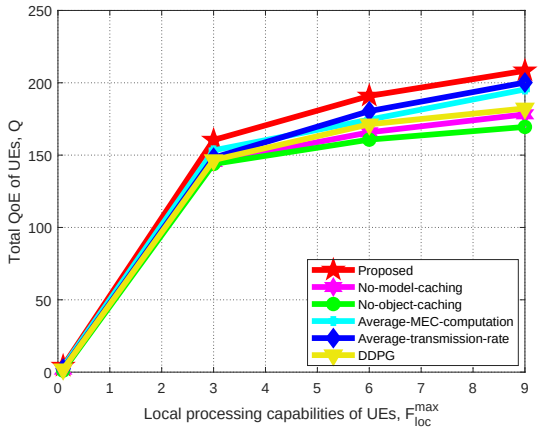

Fig. 6: The total QoE of UEs, Q vs. the local processing capabilities of the UEs, F_{max}^{loc}.

In Fig. 6, we illustrate how the total QoE of UEs changes with improvements in its local processing capacity (i.e., F_{max}^{loc}). Both the proposed algorithm and several baseline algorithms used for comparison follow this trend. When F_{max}^{loc} is as low as 0.1G, the QoE values for all algorithms are notably low. This is because limited F_{max}^{loc} forces UEs to offload foreground information to the BS, resulting in relatively large local foreground rendering delays $T_i^{f,loc}(t)$ as well as significant decoding delays $T_i^{de,off}(t)$ and $T_i^{de,loc}(t)$. Notably, in another scenario where UE processes foreground information locally, it experiences relatively large frame integration delays $T_i^{inte}(t)$. As F_{max}^{loc} increases, the overall end-to-end delay gradually decreases, leading to a continuous rise in the QoE values of all algorithms after convergence. Furthermore, it can be observed that the QoE achieved by the proposed algorithm surpasses that of all other algorithms, reaching the highest target value. This observation aligns with our earlier analysis.

In Fig. 7, we have illustrated the impact of the MEC server's maximum storage capacity(i.e., Π) on the total QoE of UEs.

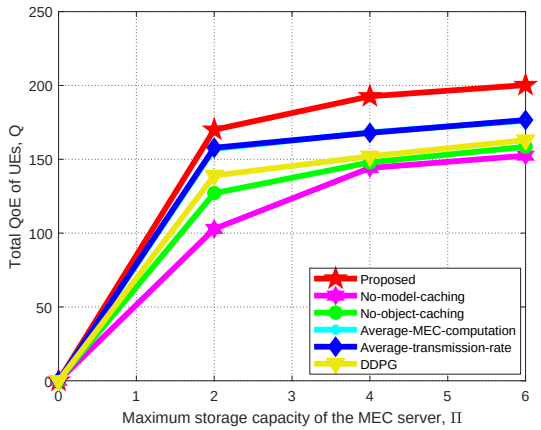

Fig. 7: The total QoE of UEs, Q vs. the maximum storage capacity of the MEC server, Π.

Since Π only plays a role in constraint (C8) of problem $(\mathcal{P}_1)$, as the algorithm converges, a larger value of Π implies that more models and objects can be cached in the BS. This reduces the necessity of downloading background information from the metaverse server, thereby decreasing the total end-to-end latency. The QoE value increases as Π increases, which is consistent with the performance trend shown in the figure. Moreover, our algorithm demonstrates superior performance compared to other benchmark algorithms under different values of Π.

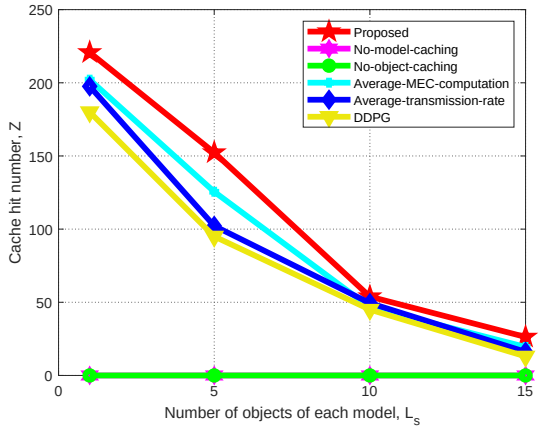

Fig. 8: Cache hit number, Z vs. the Number of objects of each model, L_s.

In Fig. 8, we depict how the number of objects of each model(i.e., L_s) impacts the cache hit number (i.e., z_i). The figure presents the numbers of cache hit for several different algorithms when $L_s = \{1, 5, 10, 15\}$. When $L_s = 1$, the proposed algorithm, Average-MEC-computation, Average-transmission-rate and the DDPG algorithms all achieve high values. This is because when each model contains only one object, the probability that this object is cached in the BS is quite high, which significantly increases the possibility of a cache hit when the UE requests this object. However, as the number of objects in the model increases, the probability that a UE's request for a specific object within the model results in a cache hit gradually decreases, which is consistent with the trend shown in the figure. For those cases where no models or objects are cached in the BS, all UE requests for models and objects must be downloaded from the metaverse server. Therefore, the cache hit rates of the No-model-caching and No-object-caching algorithms remain zero in the figure. Overall, our proposed algorithm outperforms other benchmark algorithms in terms of the cache hit number.

VII. Conclusions

In this paper, we introduced a MEC-enabled metaverse system which concludes a metaverse server, a BS and multiple mobile UEs. We aimed to an optimization problem to enhance the total QoE and the cache hit number of UEs, and reduce the energy consumption at the UE end by jointly optimizing content caching, foreground information offloading decisions and computational resources and transmission rate allocation. Then we proposed an ADRL algorithm to achieve the optimal solution by minimizing the agent's intrinsic "free energy" in combination with optimizing the specific reward. The experimental results demonstrate the proposed algorithm performs well in enhancing the total QoE and the cache hit number of UEs, and minimizing system energy consumption, and converges rapidly.

VIII. Future Work

At present, we only directly assign the value of the transmission rate to UEs through the base station. In fact, from the perspective of modeling, this is not rigorous enough. In the future, we hope to calculate the transmission rate between UEs and the BS by using the Shannon formula, and then take into account various interferences in the communication process, so as to make our model more reasonable.

In addition, in the calculation of energy consumption, we did not consider the energy consumption caused by the mobilty of UEs, which is in fact an important issue in practical scenarios and will be considered in our future work to make our model more reasonable and more in line with the actual situations.

References

[1] Y. Fu, C. Li, F. R. Yu, T. H. Luan, P. Zhao, and S. Liu, "A survey of blockchain and intelligent networking for the metaverse," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3587–3610, 2023.

[2] M. Vaezi, K. Noroozi, T. D. Todd, D. Zhao, G. Karakostas, H. Wu, and X. Shen, "Digital twins from a networking perspective," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 23 525–23 544, 2022.

[3] H. Wu, J. Chen, T. N. Nguyen, and H. Tang, "Lyapunov-guided delay-aware energy efficient offloading in iiot-mec systems," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 2117–2128, 2023.

[4] J. Shi, Y. Zhou, Z. Li, Z. Zhao, Z. Chu, and P. Xiao, "Delay minimization for noma-mmw scheme-based mec offloading," *IEEE Internet of Things Journal*, vol. 10, no. 3, pp. 2285–2296, 2023.

[5] J. Du, J. Xu, J. Jiang, Y. Zeng, R. Jin, and H. He, "Joint optimization algorithm for blockchain driven edge computing systems," *Xi'an University of Post and Telecommunication*, vol. 28, no. 6, pp. 1–11, 2023 (in Chinese).

[6] J. Du, J. Wang, A. Sun, J. Qu, J. Zhang, C. Wu, and D. Niyato, "Joint optimization in blockchain and mec enabled space-air-ground integrated networks," *IEEE Internet of Things Journal*, early access, 2024.

[7] D. Van Huynh, S. R. Khosravirad, A. Masaracchia, O. A. Dobre, and T. Q. Duong, "Edge intelligence-based ultra-reliable and low-latency communications for digital twin-enabled metaverse," *IEEE Wireless Communications Letters*, vol. 11, no. 8, pp. 1733–1737, 2022.

[8] N. Aung, S. Dhelim, L. Chen, H. Ning, L. Atzori, and T. Kechadi, "Edge-enabled metaverse: The convergence of metaverse and mobile edge computing," *Tsinghua Science and Technology*, vol. 29, no. 3, pp. 795–805, 2024.

[9] Z. Xu, Z. Yuan, W. Liang, D. Liu, W. Xu, H. Dai, Q. Xia, and P. Zhou, "Learning-driven algorithms for responsive ar offloading with non-deterministic rewards in metaverse-enabled mec," *IEEE/ACM Transactions on Networking*, vol. 32, no. 2, pp. 1556–1572, 2024.

[10] M. Cheng, Z. Su, Y. Wu, Q. Xu, M. Dai, and D. Fang, "Mec-enabled cooperative rendering in metaverse: A coalition formation game approach," in *ICC 2024 - IEEE International Conference on Communications*, 2024, pp. 4548–4553.

[11] W. Zhang, X. Chen, R. Yin, C. Wu, and Y. Cao, "Information freshness optimization in uav-aided vehicular metaverse: A ppo-based learning approach," in *ICC 2024 - IEEE International Conference on Communications*, 2024, pp. 4755–4760.

[12] O. Hashash, C. Chaccour, W. Saad, K. Sakaguchi, and T. Yu, "Towards a decentralized metaverse: Synchronized orchestration of digital twins and sub-metaverses," in *ICC 2023 - IEEE International Conference on Communications*, 2023, pp. 1905–1910.

[13] P. D. J. Schulman, F. Wolski, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[14] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[15] R. Chai, H. Niu, J. Carrasco, F. Arvin, H. Yin, and B. Lennox, "Design and experimental validation of deep reinforcement learning-based fast trajectory planning and control for mobile robot in unknown environment," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 4, pp. 5778–5792, 2024.

[16] Q. Luo and T.-P. Tan, "Rarsmsdou: Master the game of doudizhu with deep reinforcement learning algorithms," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 8, no. 1, pp. 427–439, 2024.

[17] C. Fang, Z. Hu, X. Meng, S. Tu, Z. Wang, D. Zeng, W. Ni, S. Guo, and Z. Han, "Drl-driven joint task offloading and resource allocation for energy-efficient content delivery in cloud-edge cooperation networks," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 12, pp. 16 195–16 207, 2023.

[18] S. Bi, L. Huang, H. Wang, and Y.-J. A. Zhang, "Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 11, pp. 7519–7537, 2021.

[19] L. Zhou, S. Leng, and Q. Wang, "A federated digital twin framework for uavs-based mobile scenarios," *IEEE Transactions on Mobile Computing*, vol. 23, no. 6, pp. 7377–7393, 2024.

[20] J. Du, Z. Yu, S. Aijing, J. Jiang, H. Zhao, N. Zhang, C. Wu, and F. R. Yu, "Secure task offloading in blockchain-enabled mec networks with improved pbft consensus," *IEEE Transactions on Cognitive Communications and Networking*, early access, 2024.

[21] L. Zhou, S. Leng, Q. Liu, and Q. Wang, "Intelligent uav swarm cooperation for multiple targets tracking," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 743–754, 2022.

[22] C. Fields, F. Fabrocini, K. Friston, J. F. Glazebrook, H. Hazan, M. Levin, and A. Marcian, "Control flow in active inference systemsłpart i: Classical and quantum formulations of active inference," *IEEE Transactions on Molecular, Biological, and Multi-Scale Communications*, vol. 9, no. 2, pp. 235–245, 2023.

[23] C. Fields, F. Fabrocini, K. Friston, J. F. Glazebrook, H. Hazan, M. Levin, and A. Marcian, "Control flow in active inference systemsłpart ii: Tensor networks as general models of control flow," *IEEE Transactions on Molecular, Biological, and Multi-Scale Communications*, vol. 9, no. 2, pp. 246–256, 2023.

[24] G. Oliver, P. Lanillos, and G. Cheng, "An empirical study of active inference on a humanoid robot," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 14, no. 2, pp. 462–471, 2022.

[25] J. Fang, Y. He, F. R. Yu, J. Li, and V. C. Leung, "Large language models (llms) inference offloading and resource allocation in cloud-edge networks: An active inference approach," in *2023 IEEE 98th Vehicular Technology Conference (VTC2023-Fall)*, 2023, pp. 1–5.

[26] Y. Ren, R. Xie, F. R. Yu, R. Zhang, Y. Wang, Y. He, and T. Huang, "Connected and autonomous vehicles in web3: An intelligence-based reinforcement learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 8, pp. 9863–9877, 2024.

[27] F. Guo, F. R. Yu, H. Zhang, H. Ji, V. C. Leung, and X. Li, "An adaptive wireless virtual reality framework in future wireless networks: A distributed learning approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 8514–8528, 2020.

[28] Z. Huang and V. Friderikos, "Mobility aware optimization in the metaverse," in *2022 IEEE Globecom Workshops*. Rio de Janeiro, Brazil, Dec. 2022, pp. 80–86.

[29] Z. Li, C. Yang, X. Huang, W. Zeng, and S. Xie, "Coor: Collaborative task offloading and service caching replacement for vehicular edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 7, pp. 9676–9681, 2023.

[30] Z. Zhang, C.-H. Lung, X. Wei, M. Chen, S. Chatterjee, and Z. Zhang, "In-network caching for icn-based iot (icn-iot): A comprehensive survey," *IEEE Internet of Things Journal*, vol. 10, no. 16, pp. 14 595–14 620, 2023.

[31] J. Du, J. Xu, S. Aijing, K. Jiawen, H. Ye, F. R. Yu, and V. C. M. Leung, "Profit maximization for multi-time-scale hierarchical drl-based joint optimization in mec-enabled air-ground integrated networks," *IEEE Transactions on Communications*, early access, 2024.

[32] J. Feng, L. Liu, X. Hou, Q. Pei, and C. Wu, "Qoe fairness resource allocation in digital twin-enabled wireless virtual reality systems," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 11, pp. 3355–3368, 2023.

[33] A. K. S. C. L. B. Alexander Tschantz, Beren Millidge, "Reinforcement learning through active inference," arXiv:2002.12636 [cs], 2020.

[34] A. D. Noel, C. van Hoof, and B. Millidge, "Online reinforcement learning with sparse rewards through an active inference capsule," 2021. [Online]. Available: https://arxiv.org/abs/2106.02390

[35] K. Boos, D. Chu, and E. Cuervo, "Flashback: Immersive virtual reality on mobile devices via rendering memoization," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, 2016, pp. 291–304.

[36] K. Xiang and Y. He, "Uav-assisted mec system considering uav trajectory and task offloading strategy," in *ICC 2023 - IEEE International Conference on Communications*, 2023, pp. 4677–4682.

[37] J. Zhang, L. Ding, L. Zhu, N. Cheng, and T. H. Luan, "Serial or parallel: Reverse offloading based mec-assisted joint computing," in *2023 IEEE 98th Vehicular Technology Conference (VTC2023-Fall)*, 2023, pp. 1–5.

[38] Y. Zhao, A. Xiao, S. Wu, C. Jiang, L. Kuang, and Y. Shi, "Adaptive partitioning and placement for two-layer collaborative caching in mobile edge computing networks," *IEEE Transactions on Wireless Communications*, vol. 23, no. 8, pp. 8215–8231, 2024.

[39] L. Zhou, S. Leng, Q. Wang, T. Q. Quek, and M. Guizani, "Cooperative digital twins for uav-based scenarios," *IEEE Communications Magazine*, pp. 1–7, 2024.

Jianbo Du received her Ph.D. in communication and information systems from Xidian University, Xi'an, Shaanxi, China, in 2018. She was a visiting scholar at Carleton University, Canada, in 2019. She is now an associate professor with the School of Communication and Information Engineering, Xi'an University of Posts and Telecommunications. She has published more than 70 research papers in leading journals and flagship conferences and 7 of them are ESI Top 1% highly cited papers, and her papers have been cited more than 2,900 times on Google Scholar. She was selected as the Top 2% world top scientists in 2022-2024. She hosts many projects including NSFC General Project and Youth Project. She serves as the editor of IEEE WCL and IEEE OJVT, the guest editor of IET Communications, etc. She also serves as the TPC chair, session chair, and TPC member of many conferences, and the reviewer for more than 30 leading journals. She has been awarded the excellent reviewer of IEEE TCOM, IEEE TNSE, and IEEE CL in 2021 and 2022, and the excellent reviewer of Science China - Information Science in 2023. Her research interests include mobile edge computing, blockchain, space-air-ground integrated networks, deep reinforcement learning, resource allocation, etc., and their applications in wireless communications.

Jie Gong obtained a Bachelor's degree in Electronic Information Engineering from Shanxi University in 2022. He is currently pursuing a Master's degree in Electronic Information at Xi'an University of Posts and Telecommunications. His research focuses on deep reinforcement learning algorithms, mobile edge computing, particularly their applications in wireless communication.

Xiaoli Chu (Senior Member, IEEE) received the B.Eng. degree in electronic and information engineering from Xian Jiaotong University, China, in 2001, and the Ph.D. degree in electrical and electronic engineering from The Hong Kong University of Science and Technology in 2005. From 2005 to 2012, she was with the Centre for Telecommunications Research, Kings College London, U.K. She is currently a Professor with the School of Electrical and Electronic Engineering, The University of Sheffield, U.K. She has co-authored over 200 peer-reviewed journals and conference papers, including eight ESI Highly Cited Papers and the IEEE Communications Society 2017 Young Author Best Paper. She has co-authored/co-edited four books: Fog-Enabled Intelligent IoT Systems (Springer 2020), Ultra Dense Networks for 5G and Beyond (Wiley 2019), Heterogeneous Cellular Networks-Theory, Simulation and Deployment (Cambridge University Press 2013), and 4G Femtocells: Resource Allocation and Interference Management (Springer 2013). Dr. Chu received the Best Performing Editor Award of the IEEE Vehicular Technology Society in 2024 and the IEEE Communications Letters Exemplary Editor Award in 2018. She is currently an Associate Editor of IEEE Transactions on Network Science and Engineering, IEEE Open Journal of Vehicular Technology, and IEEE Transactions on Machine Learning in Communications and Networking.

Zehui Xiong received the B.Eng. degree in telecommunications engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, and the Ph.D. degree in computer science and engineering from Nanyang Technological University (NTU), Singapore. He is currently an Assistant Professor with the Singapore University of Technology and Design (SUTD), and also an Honorary Adjunct Senior Research Scientist with the Alibaba-NTU Singapore Joint Research Institute, Singapore. His research interests include wireless communications, the Internet of Things, blockchain, edge intelligence, and metaverse.

Xianfu Chen received his Ph.D. degree (with Hons.) from the Zhejiang University, Hangzhou, China, in 2012. In 2012, he joined the VTT Technical Research Centre of Finland, Oulu, Finland, as a Research Scientist and as a Senior Scientist from 2013 to 2023. He is currently a Chief Research Engineer with the Shenzhen CyberAray Network Technology Co., Ltd, Shenzhen, China. His research interests include various aspects of wireless communications and networking, with emphasis on human-level and artificial intelligence for resource awareness in next-generation communication networks. He was the recipient of the 2021 IEEE Communications Society Outstanding Paper Award, and the 2021 IEEE Internet of Things Journal Best Paper Award. He is an Editor of IEEE Open Journal of the Communications Society, an Academic Editor of Wireless Communications and Mobile Computing, and an Associate Editor of China Communications.

Mianxiong Dong (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science and engineering from The University of Aizu, Aizuwakamatsu, Japan.,He is the Vice President and a Professor with the Muroran Institute of Technology, Muroran, Japan.,Prof. Dong is the recipient of the NISTEP Researcher 2018 from MEXT (one of only 11 people in Japan) in recognition of significant contributions in science and technology, the Hokkaido Science and Technology Incentive Award 2019, and the Young Scientists' Award 2021 from MEXT. He serves as a Program Officer of JST Support for Pioneering Research Initiated by the Next Generation. He is a Clarivate Analytics 2019 and 2021 Highly Cited Researcher (Web of Science), and a Foreign Fellow of the Engineering Academy of Japan.

F. Richard Yu (Fellow, IEEE) received the Ph.D. degree in electrical engineering from The University of British Columbia in 2003. From 2002 to 2006, he was with Ericsson, Lund, Sweden, and a start-up in California, USA. He was also with Carleton University, Ottawa, ON, Canada, in 2007, where he is currently a Professor. His research interests include cross-layer/cross-system design, security, green ICT, and QoS provisioning in wireless-based systems. Prof. Yu received the IEEE Outstanding Service Award in 2016, the IEEE Outstanding Leadership Award in 2013, the Carleton Research Achievement Award in 2012, the Ontario Early Researcher Award (formerly Premiers Research Excellence Award) in 2011, the Excellent Contribution Award at IEEE/IFIP TrustCom 2010, the Leadership Opportunity Fund Award from Canada Foundation of Innovation in 2009, and the Best Paper Awards at IEEE ICC 2014, Globecom 2012, IEEE/IFIP TrustCom 2009, and Internal Conference on Networking 2005. He has served as the TPC co-chair for numerous conferences. He is a registered Professional Engineer in the province of Ontario, Canada. He serves as the Vice-Chair for the IEEE Technical Committee on Green Communications and Computing and a member of Board of Governors for the IEEE Vehicular Technology Society.