QoS Optimization via Computation Offloading in Metaverse Environment

Zhiyuan Ge[1,2], Pengcheng Zhang[1,2,*], Huiying Jin[3], Hai Dong[4], Shunhui Ji[1,2], Jiajia Li[1,2], and Qi Wang[1,2]

[1]Key Laboratory of Water Big Data Technology of Ministry of Water Resources, Hohai University, Nanjing, China
[2]College of Computer Science and Software Engineering, Hohai University, Nanjing, China
[3]School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, China
[4]School of Computing Technologies, RMIT University, Melbourne, Australia
{*zhiyuange, pchzhang, shunhuiji, jiajiali, qiwang*}*@hhu.edu.cn, hyjin@njupt.edu.cn, hai.dong@rmit.edu.au*

Abstract—**The emergence of the metaverse signifies a paradigm shift in Internet technology, offering a comprehensive virtual social platform spanning various domains such as social interaction, gaming, healthcare, and tourism. This new era of the metaverse is facilitated by advancements in next-generation digital technologies including edge computing, artificial intelligence, virtual reality, augmented reality, and blockchain. In the metaverse, the quantity and variety of services requested by users may surpass those in other environments, and existing work cannot be applied to metaverse QoS (Quality of Service) optimization. To address this problem, this paper proposes Meta-PPO, an optimization method for enhancing the QoS of metaverse services using reinforcement learning. Firstly, metaverse services are categorized into virtual scene services and meta-services, providing a comprehensive framework for analysis. Secondly, Meta-PPO, based on the proximal policy optimization algorithm, is introduced to optimize the QoS of metaverse services. This method effectively balances the objectives of minimizing average delay and maximizing resource utilization of mobile devices by making informed offloading decisions for the identified service categories. Simulation results demonstrate the superiority of the proposed method over existing techniques, showcasing its suitability and effectiveness for enhancing the QoS of metaverse service.**

Index Terms—**Metaverse, Computation offloading, Edge computing, Reinforcement learning, Quality of Service**

I. Introduction

The metaverse originated from the novel "Snow Crash" by American science fiction writer Neal Stephenson [1]. This work is considered one of the earliest explorations of how the virtual and real worlds interact, and to some extent reflects the impact of virtual worlds on social structures and ideologies. "World of Warcraft" and "Second Life", two online games, are early examples of metaverse implementation [2]. These video games offer a simulated world in which users can create virtual avatars, communicate with other users, explore virtual environments, undertake quests, and engage in social activities. During a global live event, Facebook CEO and founder Mark Zuckerberg announced that the company would transition into a "Metaverse company" and develop a metaverse on its social media platform, integrating augmented reality (AR) and virtual reality (VR) [3]. The metaverse is poised to become a significant force in the upcoming Internet era, influencing various aspects of our lives.

Quality of Service (QoS) serves as one of the non-functional attributes of a service, providing a benchmark for selecting services. To ensure the satisfaction of users within the metaverse, numerous research endeavors have delved into optimizing the QoS of metaverse services. As a real-time virtual interactive platform, the delay and energy consumption of metaverse services are widely recognized as key metrics affecting users' Quality of Experience (QoE). A series of methods have been proposed to reduce delay and energy consumption [4]–[8]. However, their attention has primarily been directed towards the differences in QoS metrics between metaverse services and other services, without delving into the fundamental characteristics within metaverse services that contribute to their heightened sensitivity to QoS. Additionally, optimization efforts have commonly been concentrated on specific categories of services within the metaverse. For instance, Zhang et al. [9] and Huang et al. [10] have discussed AR services within the metaverse, while Liu et al. [8] and Meng et al. [11] have examined VR services. While these studies have demonstrated enhancements in user QoE for specific services, such optimization approaches may result in incomplete optimization of metaverse services as a whole. As an emerging service paradigm, metaverse services are required to more comprehensively address the diverse requirements of both users and providers.

Despite these advancements, there are still two major gaps in current approaches:

- **Concentrating exclusively on optimizing individual services within metaverse environments can result in incomplete optimization strategies.** Such approaches may fail to consider the unique characteristics of the metaverse, where users are not bound by the temporal and spatial constraints of the physical world. In the metaverse, the demand for various service types within virtual scenes can far exceed what is typically observed in real-world scenarios. A single metaverse service requested by a user may encompass multiple services with diverse

* Corresponding Author.

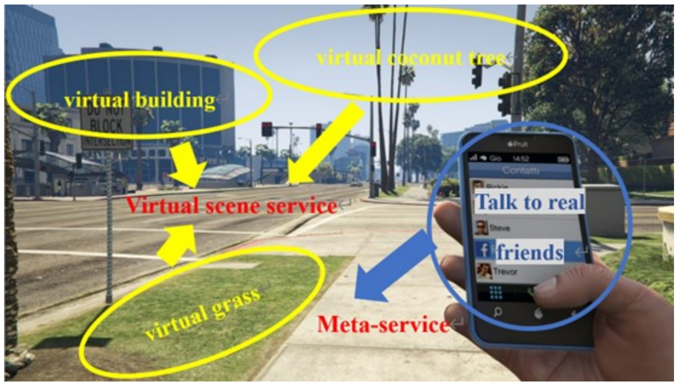


Fig. 1. The example of metaverse service

functionalities, mirroring the complexities of the virtual environment. Focusing solely on optimizing a limited number of service types may prove insufficient in meeting the genuine needs of metaverse users.

- **Previous methods overlooked the resource utilization of mobile devices.** Current development trends indicate that mobile devices, including head-mounted displays (HMD), will become essential elements of the metaverse. Compared to other environments, the dependence of metaverse users on mobile devices will increase significantly. Facing the inevitable consumption of mobile device resources, the challenge is how to better utilize mobile devices to enhance the user experience in the metaverse. Previous QoS optimization efforts primarily focused on minimizing the energy consumption of services. However, in the metaverse, energy savings alone cannot directly improve the user experience. In addition to energy consumption, the utilization of other mobile device resources has received scant attention. Overlooking the utilization rate of mobile device resources will have a significant impact on metaverse users.

Consider the following example illustrated in Fig.1: User A wears an HMD and enters the metaverse game X, starting in a rainforest environment. Initially, A's requested metaverse service is to render and construct the rainforest scene. However, as A encounters challenges within the game, they opens a web browsing interface within the metaverse to search for game guides. At this moment, A's requested metaverse service shifts to search services. Meanwhile, A's friend B sends a voice request, inviting A to engage in a voice call while remaining within the rainforest scene. Here, A's requested metaverse service transitions to voice services. Subsequently, A accepts B's invitation to explore a virtual Forbidden City scene together resulting in a new request for virtual scene services. Upon arrival at the entrance of the Forbidden City, A requests navigation services, which manifest directly beneath the user's feet. Based on the aforementioned example, this paper categorizes metaverse services and proposes a QoS optimization method for metaverse services based on reinforcement learning. The contributions of this paper are as follows:

- We analyzed metaverse services and proposed a novel classification scheme. We suggested categorizing metaverse services into two main types: virtual scene services and meta-services. Virtual scene services are utilized to construct virtual scenes within the metaverse (e.g., rendering rainforest scenes, rendering Forbidden City scenes), while meta-services cater to users' diverse needs (e.g., search services, voice services, navigation services) within the metaverse. This categorization method is applicable to services within the metaverse.
- We designed a QoS optimization method using the proximal policy optimization algorithm from reinforcement learning. In selecting QoS metrics, we considered delay as well as the energy consumption and memory usage of mobile devices. Unlike previous studies that aimed to minimize the energy consumption of services, we reframed energy consumption and memory concerns to enhance the utilization efficiency of mobile devices, aligning with the reality of limited resources on head-mounted displays. This approach achieves a balance between minimizing average delay and maximizing utilization of mobile resources by simultaneously making offloading decisions for both types of services.
- QoS data for virtual scene services were obtained through simulation and emulation. Additionally, QoS data for meta-services were gathered from publicly available datasets. Subsequently, we compiled datasets that adhere to the specifications of the metaverse service settings proposed in our study by combining virtual scene services and meta-services in varying proportions. Using the collected dataset, we conducted a series of experiments to demonstrate the effectiveness of our proposed approach. Simulation results demonstrate that our approach outperforms traditional methods.

The remainder of this paper is organized as follows: Section II introduces related work. Section III describes the details of the metaverse model. Section IV provides an introduction to the optimization method. Section V presents the experimental results. Finally, Section VI concludes the paper.

II. Related Work

A. *Optimization for QoS in Cloud and Edge environments*

With the increasing development of cloud computing and edge computing, QoS optimization has become a current research hotspot. In the cloud computing environment, Zhao et al. [12] proposed a strategy to improve users' QoS through the collaboration of local clouds and Internet clouds. Zhang et al. [13] introduced a theoretically optimal mobile cloud computing framework for energy consumption under stochastic wireless channels, which adjusts CPU frequency savings through different offloading modes. In addition to mobile energy consumption, Ge et al. [14] presented a game theory approach to optimize the energy consumption of the entire cloud computing system. Chen et al. [15] designed a multi-user distributed computing offloading algorithm that achieves

Nash equilibrium and demonstrated the effectiveness of their algorithm through experiments. Considering that the operation of most cloud services is primarily dependent on infrastructure servers [16], there is a need for unique methods in the metaverse environment to consider factors such as resource constraints and execution delay of mobile devices. Consequently, the QoS optimization methods applicable in cloud environments are not suitable for the metaverse.

Unlike the cloud environment, mobile edge computing can bring services closer to users to improve QoS. Xiao et al. [17] aimed to minimize the maximum delay of the system. They decomposed the problem into task offloading and transmission power allocation and solved them separately using matching theory and heuristic ideas. Shu et al. [18] proposed a coordinated EFO algorithm that balances competition among users for wireless communication and computing resources. By considering the dependency relationships of sub-tasks in DAGs, their method reduces overall system delay. You et al. [19] addressed the multi-user resource allocation problem by formulating it as a convex optimization problem. They proposed a low-complexity algorithm to minimize mobile energy consumption while meeting delay constraints. Song et al. [20] proposed an approach to enhance task management in edge computing networks, aiming to address resource limitations and network challenges in IoT infrastructures while improving task processing capabilities and meeting QoS requirements. Apart from the well-known disparities in QoS and differences in quantity, users are unable to concurrently request multiple services with distinct functionalities in the physical world. However, in the metaverse, where there are no limitations of time and space, the integration of services with diverse functionalities and categories is required to offer a unified service to users. There is currently no edge-based QoS optimization approach specifically designed for concurrently optimizing services with distinct functionalities or categories. Consequently, existing edge-based QoS optimization approaches are incapable of effectively optimizing the multi-layered services required within the metaverse.

In summary, QoS optimization methods in cloud and edge environments are difficult to directly apply to the metaverse environment.

B. Optimization for QoS in Metaverse

Xu et al. [21] presented a broad definition of metaverse services while also discussing the key technologies supporting metaverse services, as well as the challenges and opportunities for metaverse services in the future. Currently, a predominant approach adopted by many researchers is to treat supporting devices for the metaverse, such as smart glasses, as local devices within the MEC environment. Gao et al. [22] proposed an algorithm named MO-SAC, which utilizes a dual-level central controller to localize users within the metaverse. This algorithm jointly optimizes transmit power, RIS phase shift, and error decoding probability to achieve the simultaneous minimization of service cost and transmission delay in metaverse services. Huang et al. [23] investigated the Mixed Augmented Reality (MAR) services within the metaverse and proposed a resource allocation framework tailored for metaverse MAR services. Sahraoui et al. [24] proposed an edge hybrid computing architecture that leverages the distributed computing paradigm supported by the edge to address the substantial computational costs required by metaverse applications. Through simulation-based verification, they demonstrated that their architecture can reduce delay by 50% compared to traditional cloud-based metaverse applications. Zhang et al. [9] incorporated the consideration of location information for AR services in metaverse applications. Upon each user being localized, they communicate with MEC servers. These servers adaptively adjust the resolution of AR services while balancing the QoE for users. Chu et al. [25] assert that metaverse applications demand unprecedented amounts of resources. They proposed an application grouping solution called MetaInstance, which aims to address the allocation of a significant number of diverse resources. Through experimentation, they demonstrated the superiority of their approach.

In general, most of the current studies regard metaverse services as services with higher resource demand and more sensitive QoS values. These studies often focus on optimizing singular services like VR, AR, and MAR within metaverse services. However, the metaverse typically encompasses a variety of services, and optimizing only a single service is insufficient to meet the diverse needs of metaverse users. Moreover, existing works primarily focus on reducing energy consumption from a service perspective, while overlooking the significance of mobile devices in the metaverse. Therefore, this paper discusses a novel approach for categorizing metaverse services and proposes a method utilizing deep learning algorithms to optimize the QoS of metaverse services and the resource utilization of mobile devices.

III. System Overview

In this section, firstly, we introduce a novel classification method for metaverse services. Secondly, we present the metaverse model employed in our study and define the metrics of task delay and mobile device resource utilization. Finally, we formulate the optimization problem based on the proposed approach.

A. Metaverse Service

In the metaverse, users are unbounded by temporal and spatial constraints, rendering the selection of service types unpredictable. Furthermore, services with lower interdependencies in the physical realm are highly likely to be simultaneously requested in the metaverse environment. If optimization efforts are limited to a specific subset of services, it would be challenging to adequately address the users' demands. Thus, in this paper, we propose a novel definition for metaverse services by classifying them into two categories: virtual scene services and meta-services, as represented by the following equation:

Metaverse Service = Virtual Scene Service + Meta-Service

1) Virtual Scene Service: When users enter the metaverse by wearing devices such as head-mounted displays, they are immersed in a completely new virtual world. This virtual world is constructed by combining one or more virtual scenes, which can resemble real-world settings such as movie theaters, gyms, or conference rooms. It can also feature game-like environments that do not exist in the physical world, such as Mars bases, deserts, or rainforests. These scenes are predominantly designed and provided by metaverse service providers, with only a small portion being user-designed and cached within metaverse applications.

2) Meta-Service: Meta-services, on the other hand, refer to the "functional services" that users actually utilize within the metaverse. These services are similar to those found in traditional environments and are provided by service providers to users. They offer various functionalities and services, such as data querying, business logic processing, and file transfer. They can span across different platforms and technologies, enabling interoperability between different applications. Common types of meta-services include video services, search services, and more.

Compared to previous methodologies, our proposed novel categorization approach offers a higher-dimensional framework for defining metaverse services. By categorizing services according to the unique characteristics of the metaverse virtual world, our method encompasses various service categories within the metaverse. By optimizing QoS of metaverse services based on our categorized approach, we can achieve comprehensive optimization targeting multiple service categories, rather than individually optimizing specific categories based on traditional environments. The outcomes of our categorization method effectively resolve the issue of incomplete optimization in metaverse services.

B. Metaverse Model

In this paper, we focus on a metaverse system where there is a single metaverse user with a mobile device (MD). The system includes a mobile base station (BS) equipped with a server, which can provide services to the MD. Our system operates in a time slot manner, where each user request for metaverse services generates a series of pending tasks on the MD. Each time slot is dedicated to completing one task. Due to the limited resources of the MD (e.g., memory, energy), certain tasks that are executed locally may not fully meet the user's requirements, Therefore, a portion of the tasks is offloaded to the BS to alleviate resource constraints. we consider the computational and storage capabilities of both the BS and MD, $BS = (F_{bs}, C_{bs}, B_{bs}, M_{bs}), MD = (F_{md}, C_{md}, B_{md}, M_{md}, E_{md})$, F and C respectively denote the frequency (GHz) and the number of cores of the CPU, B denotes available bandwidth resources (Gbps), M denotes available memory resources (bits), and E denotes energy consumption (J). In our system, the energy consumption and memory of MD are limited, and insufficient to support user requests for all services. However, given that in practical scenarios, the memory capacity of BS does not affect whether tasks are offloaded when targeting a single user, we approximately regard the memory of BS as infinite.

C. Task Model and Optimization Metrics

In this section, we will introduce two selected optimization metrics in the metaverse system: delay and mobile device resource utilization. It is important to note that tasks in the task list can be executed concurrently. Therefore, the system will allocate resources equally among tasks based on offloading decisions.

1) Delay: Firstly, in our system, as users request metaverse services, a series of tasks, denoted as $Task_n^s = (I_n, O_n, P_n)$. Here, n represents the ID of the task, s represents the offloading decision result of the task, I_n represents the input data size of the task, O_n represents the instruction count size, and P_n represents the task type. We define the offloading decision for the computational tasks T_J generated in the first J time slots on the MD as $S_J = (s_1, s_2, s_3 \dots s_j)$:

$$S_j = \begin{cases} 0, \text{Local computing} \\ 1, \text{Offloading to the server} \end{cases} \tag{1}$$

Considering the substantial disparity in resources between the local device and the server, different offloading decisions have a significant impact on the delay of tasks. Below, we delineate the differences in task delay D_n^s under two distinct offloading decision strategies:

- **Local computing:** When $s_j = 0$, it indicates that the task in the jth time slot is decided to be executed locally. Therefore, in this case, the task delay only includes the computation delay $d_{cal,n}^0$, where $\text{F}(\cdot) = \frac{F_{md} * C_{md}}{Num(Task_n^0)}$. Here, $Num(Task_n^0)$ represents the number of tasks executed locally under the current decision. The formula is as follows:
$$d_{cal,n}^0 = \frac{Task_n^0(O_n)}{F(F_{md}, C_{md})} \tag{2}$$
- **Offloading to the server:** When $s_j = 1$, it indicates that the task in the jth time slot is decided to be offloaded to the server for execution. Therefore, in this case, the task delay includes both the transmission delay and the computation delay on the server, i.e., $d_n^1 = d_{cal,n}^1 + d_{tran,n}^1$. Here, $\text{G}(\cdot) = \frac{F_{bs} * C_{bs}}{Num(Task_n^1)}$, where $Num(Task_n^1)$ represents the number of tasks offloaded to the edge server for execution under the current decision. The specific formula is as follows:
$$d_{cal,n}^1 = \frac{Task_n^1(O_n)}{G(F_{bs}, C_{bs})} \tag{3}$$
Similar to [26], when tasks are offloaded to the edge server for execution, we consider that the edge server has sufficient resources and the output data returned from the edge server to the mobile device is typically small and can be ignored compared to the input data. Therefore, our transmission delay is calculated only from the mobile device to the server, as follows:
$$d_{tran,n}^1 = \frac{Task_n^1(I_n)}{G(B_{bs}, 1)} \tag{4}$$

2) Resource Utilization: Mobile devices with constrained resources have a significant impact on the sustained QoS of metaverse services. In offloading scenarios, it is a common practice to evenly allocate CPU resources to tasks based on decision results [27]. Therefore, with relatively stable CPU resources, we have selected two attributes, energy consumption and memory, which exhibit variations depending on different offloading decisions. We define the utilization rate of these two attributes, weighted together, as the resource utilization rate of the mobile device. This serves as our second optimization objective.

a) Energy: As time progresses, mobile devices consume varying amounts of energy due to different offloading decisions. E_0^{device} represents the initial energy consumption of the mobile device. We calculate the remaining energy E_j^{device} of the mobile device for each time slot and the energy E_n^{Task} required for task $Task_n^s$ to be executed locally in the current time slot. Additionally, our study aims to explore the impact of different offloading methods on mobile devices. Therefore, we neglect energy consumption during idle times of the mobile device, i.e., when $s_j = 1$.

$$E_j^{device} = E_0^{device} - \sum_{n=0}^{j-1} E_n^{Local} \tag{5}$$

$$E_n^{Local} = \begin{cases} 0 & , \quad s_j = 1 \\ E_n^{Task} & , \quad s_j = 0 \end{cases} \tag{6}$$

$$E_n^{Task} = \mu * d_n^0 \tag{7}$$

Here, μ is a constant representing the energy consumption coefficient per unit time for the mobile device hardware. It is important to note that when $E_j^{device} < E_j^{Task}$, it means that the mobile device cannot meet the energy consumption required for executing the next task locally. In such cases, the task in the next time slot can only be offloaded to the server.

b) Memory: Similar to energy consumption, once a task is decided to be computed locally, the memory of the mobile device gradually gets occupied by the input and instruction data of the locally executed tasks. M_0^{device} represents the initial memory of the mobile device. When the remaining memory of the mobile device, M_j^{device}, is not sufficient to accommodate the memory requirement of the task in the current time slot, M_j^{Task}, the task will be offloaded to the server. The specific formula is as follows:

$$M_j^{device} = M_0^{device} - \sum_{n=0}^{j-1} M_n^{Local} \tag{8}$$

$$M_n^{Local} = \begin{cases} 0 & , \quad s_j = 1 \\ Task_n^s(I_n) + Task_n^s(O_n) & , \quad s_j = 0 \end{cases} \tag{9}$$

After all the tasks have been assigned offloading decisions, the resource utilization rate R can be defined as follows:

$$R = \alpha * \frac{E_0^{device} - E_j^{device}}{E_0^{device}} + \beta * \frac{M_0^{device} - M_j^{device}}{M_0^{device}} \tag{10}$$

Where α and β represent the weights for energy and memory, respectively, satisfying $\alpha + \beta = 1$.

D. Optimization Problem

According to the aforementioned system model, we have set both task delay and mobile device resource utilization rate as optimization objectives. It is anticipated that increasing the device resource utilization rate implies executing more services locally. However, MDs have limited resources, and allocating more tasks locally is likely to result in increased delay. Therefore, our two optimization objectives are likely to be conflicting. Specifically, our optimization problem can be formulated as follows:

$$\textit{MOP: Min } q1 = \sum_{n=1}^{j} D_{cal,n}^{s} + \sum_{n=1}^{j} D_{tran,n}^{s} \tag{11}$$

$$Max{:}q2 = R = \alpha * \frac{E_0^{device} - E_j^{device}}{E_0^{device}} + \beta * \frac{M_0^{device} - M_j^{device}}{M_0^{device}} \tag{12}$$

$$\text{s.t.} \quad C_1 : \alpha + \beta = 1 \tag{13}$$

$$C_2 : \sum_{n=0}^{j} E_n^{Local} \leq E_0^{device} \tag{14}$$

$$C_3 : E_j^{device} \geq E_j^{Task} \tag{15}$$

$$C_4 : \sum_{n=0}^{j} M_n^{Local} \leq M_0^{device} \tag{16}$$

$$C_5 : M_j^{device} \geq M_j^{Task} \tag{17}$$

C_1 represents the adjustability of energy consumption and memory within resource utilization through weighting. Furthermore, C_2 and C_4 stipulate that the energy consumption E_n^{Local} and memory M_n^{Local} utilized by locally executed tasks should not surpass their initial levels. Concurrently, C_3 and C_5 dictate that the current remaining energy consumption E_j^{device} and memory M_j^{device} on the mobile device must satisfy the requirements for executing the Jth time slot task locally.

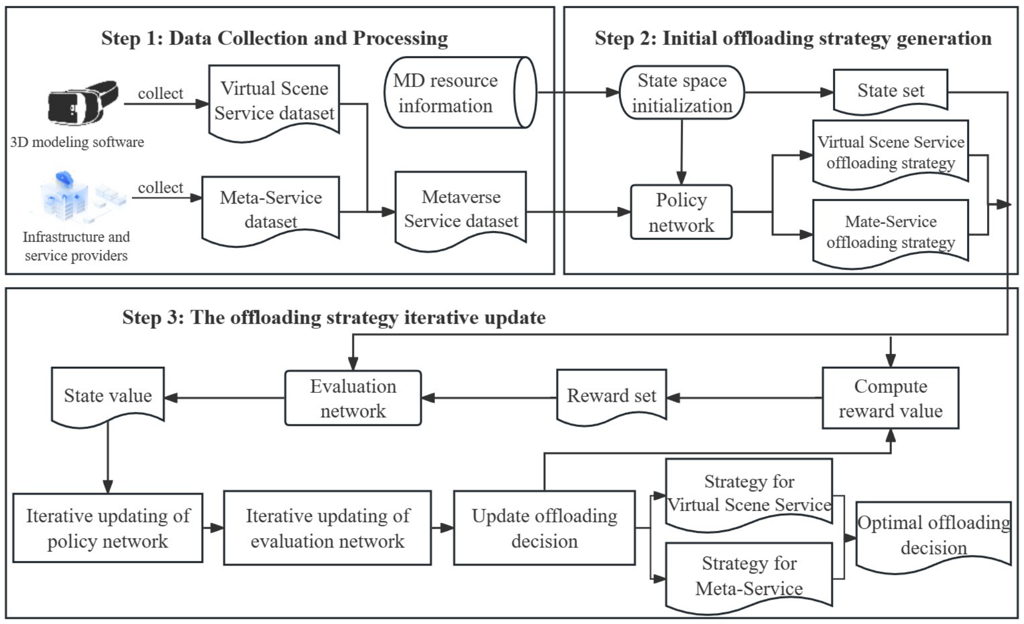
IV. The Meta-PPO Approach

Considering that reinforcement learning can make sequential decisions tailored to the problem while considering the long-term benefits brought by the decisions [28], we propose an optimization method for the QoS of metaverse services called Meta-PPO. The overall framework of the Meta-PPO is outlined below.

A. Method Overview

The overall framework of the proposed optimization method is illustrated in Fig. 2. The method comprises the following three steps:

- *Data collection and processing*. The data collected in this paper consists of QoS data for metaverse services. QoS data comprises values of multiple QoS attributes corresponding to virtual scene services and meta-services delineated by the metaverse service. After cleaning out anomalies and invalid data, the collected data is aggregated and combined in diverse proportions and quantities to accurately reflect the service demands of metaverse



Step 1: Data Collection and Processing
collect
3D modeling software
Virtual Scene Service dataset
MD resource information
collect
Infrastructure and service providers
Meta-Service dataset
Metaverse Service dataset
Step 2: Initial offloading strategy generation
State space initialization
State set
Virtual Scene Service offloading strategy
Policy network
Mate-Service offloading strategy
Step 3: The offloading strategy iterative update
State value
Evaluation network
Reward set
Compute reward value
Iterative updating of policy network
Iterative updating of evaluation network
Update offloading decision
Strategy for Virtual Scene Service
Strategy for Meta-Service
Optimal offloading decision

Fig. 2. Overall workflow diagram of Meta-PPO

Algorithm 1 Meta-PPO Task Offloading Algorithm

```
1: Initialize Mobile Device and Base Station
2: Initialize Buffer $memory$
3: Initialize the policy $\pi$ and optimal solution $\varphi = \{\}$
4: Initialize learning rate of actor and critic
5: **for** episode=1,2,...,E **do**
6:     Set $\pi \rightarrow \tilde{\pi}$ and the solution $\varphi = \{\}$
7:     **for** time slot $t = \mathcal{T}$ **do**
8:         Initialize the state $s$
9:         Select the action a with the policy $\tilde{\pi}$
10:        Confirm a with the state s
11:        Observe the reward $r(s, a)$with action $a$
12:        Get the next state $\tilde{s}$
13:        Store $(s, a, r(s, a), \tilde{s})$ into the buffer
14:        Add the strategy to $\varphi$
15:    **end for**
16: **end for**
17: Clean the replay buffer
**Output:** The optimal solution
```

users across different virtual environments, ultimately resulting in the generation of the metaverse service dataset.

- *Initial offloading strategy generation.* Following the principles of reinforcement learning algorithms, this paper defines a policy network to generate actions for the reinforcement learning problem, namely the task offloading strategy under the context of this paper's problem.
- *The offloading strategy iterative update.* After obtaining the task offloading strategy, the corresponding QoS values (such as delay, and energy consumption) and local device resource utilization rate can be calculated based on the offloading strategies generated for each service. By adjusting the weights of the two optimization objectives, a multi-objective optimal solution can be formed.

B. Method Details

1) Data Collection and Processing: Considering the previously defined metaverse services consisting of virtual scene services and meta-services, the dataset collected in this study is composed of two sub-datasets. The first sub-dataset is the virtual scene service dataset, which contains QoS data information for constructing virtual backgrounds in the metaverse. Each QoS sample includes multiple QoS attribute values for the respective service. To capture authentic data for metaverse services, we utilized the 3D modeling software to build virtual scenes and collect the required QoS attributes for virtual scene services. The second sub-dataset is the meta-service dataset. Meta-services in the metaverse are similar to services in traditional environments, provided by service providers to fulfill various user requirements. For constructing the meta-service dataset, we collected a large amount of QoS information for different types of services from publicly available datasets. The QoS attributes included in this dataset are consistent with those collected in the virtual scene service dataset. After removing invalid QoS sample information from both sub-datasets, multiple metaverse service datasets are constructed by combining them in different proportions.

2) Initial offloading strategy generation: This paper utilizes a policy network to generate the initial offloading strategy. Based on the data collected in the first step of dataset collection, we initialize the state information and then input the state information and action dimension into the policy network to obtain the initial unloading policy. Details regarding states, actions, and others will be introduced in step three. Our policy network consists of three fully connected layers. The first layer maps the input state data s to a hidden layer H. The second layer maps the output of the hidden layer back to the hidden layer itself, thus facilitating information transmission and transformation, i.e., $H^{'} = f(H)$. The final layer maps the output of the hidden layer $H^{'}$ to the dimension of the action space, thereby obtaining the actions A that the agent may take given a certain state, i.e., $A = g(H^{'})$. We choose the hyperbolic tangent function tanh as the activation function of the policy network. To prevent common problems such as gradient explosion in reinforcement learning, we initialize the weight parameters of the neural network through orthogonal initialization.

3) The offloading strategy iterative update: After generating the initial offloading strategy, we iteratively update the offloading strategy by updating the policy network and the evaluation network through the proximal policy optimization algorithm, ultimately obtaining the optimal solution. Below, we will provide a detailed introduction to our algorithm:

The Proximal Policy Optimization (PPO) algorithm is a policy gradient-based reinforcement learning algorithm. The process of solving problems in reinforcement learning is typically modeled as a Markov Decision Process (MDP). An MDP is commonly represented by a quadruple $< S, A, P, R >$, where:

- *state*: The set of all possible states in the environment
- *action*: The set of all feasible actions available to the agent
- *transition probability*: $P^{a}_{ss'} = P(S_{t+1} = s' | S_t = s, A_t = a)$
- *reward*: The reward obtained by the agent when it takes the corresponding action in the t-th time slot.

The expected value of the reward after N episodes can be estimated as follows:

$$\bar{R}_\theta = \sum_{\tau} R(\tau)P(\tau \mid \theta) \approx \frac{1}{N}\Sigma_{n=1}^{N} R(\tau^n) \tag{18}$$

The traditional policy gradient algorithm is a method that computes the gradient of the current policy through interaction between the agent and the environment, aiming to update the current policy and ultimately find the optimal policy. The optimal policy can be represented as follows:

$$\theta^* = argmax_\theta \bar{R}_\theta \tag{19}$$

Next, we will introduce the Meta-PPO method, which is based on reinforcement learning, for optimizing QoS in the metaverse.

a) ***State Space:*** In this paper, each state within the reinforcement learning framework is characterized by the observed actions undertaken by the agent. Considering the optimization objectives set in this paper, we regard the mobile device as the agent, making unloading decisions for only one task during each time slot interval. Specifically, the state $S_j$at time slot j is denoted as $S_j = [e_j, m_j]$, where e_j represents the remaining energy of the mobile device during time slot j, and m_j signifies the remaining memory of the mobile device at the same time slot.

b) ***Action Space:*** The action space set A encompasses the entirety of feasible actions. We endeavor to optimize the specified objectives by making unloading decisions for metaverse services. Additionally, as the environmental conditions fluctuate, the feasible action set $A_j \in A$ may undergo alterations. Specifically, $a_j = 0$ corresponds to the decision to locally execute the task during the current time slot, while $a_j = 1$ indicates the decision to offload the task to the edge server for computation.

$$a_j = \begin{cases} 0, \text{Local computing} \\ 1, \text{Offloading to the server} \end{cases} \tag{20}$$

It is imperative to emphasize that if the state S_j fails to meet the resource and energy demands of the task scheduled for the subsequent time slot j, the action A_t is set to 1, denoting the necessity to offload the task to the edge server for computation.

c) ***Reward:*** For each time slot, making an appropriate unloading decision for the corresponding task is regarded as selecting an action within the current environmental state. Each action generates a specific reward. In line with the multi-objective optimization problem previously outlined, the reward at this stage is represented as an instantaneous vector. In this study, we define the reward as an array r(s_j, a_j) where r$(s_j, a_j) = [r_1(s_j, a_j), r_2(s_j, a_j), r_3(s_j, a_j)]$ Here, the reward r_1 represents the total delay for all services after executing the unloading decision during the current time slot j:

$$r_1(s_j, a_j) = -\sum_{n=1}^{j} \frac{D^s_n}{num} Num \tag{21}$$

To balance the latency of two columns of services and mitigate instances within 21 where sacrificing one service type for the sake of reducing overall delay occurs, we have introduced an additional reward term, denoted as r_2, specifically aimed at incentivizing delay reduction. The r_2 denotes the cumulative delay for virtual scene services and metaservices following the execution of the unloading decision during time slot j, given by:

$$r_2(s_j, a_j) = -(\frac{\sum\limits_{n=1}^{j} D^s_{n,vir}}{num^j_{vir}} Num_{vir} + \frac{\sum\limits_{n=1}^{j} D^s_{n,meta}}{num^j_{meta}} Num_{meta}) \tag{22}$$

Where $D^s_{n,vir}$ and $D^s_{n,meta}$ denote the delay for virtual services and meta-services, respectively. num^j_{vir} and num^j_{meta} represent the quantities of virtual services and meta-services that have been decided upon at the current time slot j, while Num_{vir} and Num_{meta} denote the total numbers of virtual services and meta-services, respectively.

According to 10, we set the reward r_3 as the utilization of the mobile device's memory and energy resources in the current time slot:

$$r_3(s_j, a_j) = \alpha * \frac{E_0^{device} - E_j^{device}}{E_0^{device}} + \beta * \frac{M_0^{device} - M_j^{device}}{M_0^{device}} \tag{23}$$

In conclusion, we can tailor various requirements by assigning distinct weights to r(s_j, a_j).

V. Experimental Evaluation

In this section, a series of experiments were meticulously devised to validate our proposed method for optimizing QoS in the metaverse. The experiments were meticulously conducted on a computer equipped with an Intel(R) Core(TM) i9-12900H processor operating at 2.50 GHz, 16.0GB of RAM, running the Windows 11 operating system, and featuring an NVIDIA GeForce RTX 3060 GPU. Our method was meticulously developed and implemented using the Python programming language and the PyTorch framework. The primary objective of our experiments is to address the following inquiries:

- **RQ1**: Does Meta-PPO satisfy the minimum delay requirements in the metaverse environment?
- **RQ2**: Does Meta-PPO outperform other reinforcement learning methods?
- **RQ3**: How does Meta-PPO achieve the optimal trade-off between delay and resource utilization?

For the metaverse system under consideration for applicable metaverse services, we focus on a scenario involving only one server and one mobile device. The CPU frequency of the mobile device is set to 3 GHz with 2 cores, and the bandwidth is set to 6 Gbps. To simulate the limited resources of the mobile device, we dynamically adjust the resources of the mobile device to be correlated with the number of services in the selected dataset. As the number of services in the dataset increases, the energy and memory of the mobile device also increase accordingly. Regarding the server, we set the CPU frequency to 30 GHz with 8 cores, and the bandwidth is set to 10 Gbps. Our algorithm parameters are shown in the following table:

TABLE I
ALGORITHM PARAMETERS

parameters	value
Learning rate of actor	1*10^(-3)
Learning rate of critic	1*10^(-3)
Clipping parameter	0.2
Number of traing round	500
Discount factor	0.9

A. Dataset Description

The dataset in this paper comprises two parts: the virtual scene service dataset and the meta-service dataset.

Given the current developmental stage of metaverse applications, the provision of virtual scenes to support users is temporarily limited. Therefore, virtual scenes exhibit a high degree of predictability. In the process of constructing the virtual scene dataset, we utilized Fusion 360 software for rendering to create 50 distinct virtual scenes to simulate scenes within the metaverse. Monitoring software was employed to record various QoS metrics such as start time, end time, data size, and required CPU resources for each virtual scene. Tasks associated with these virtual scene services were designated as type 1, and all data information was subsequently stored to compile the virtual scene service dataset.

Meta-services, similar to conventional services, are characterized by smaller data volumes and lower resource requirements such as CPU and bandwidth compared to the virtual scene services defined in this study. For the construction of the meta-service dataset, a random selection of 500, 800, 1000, 2000, and 3000 data entries was made from the Alibaba Cluster Data V2017 dataset, which was publicly released by Alibaba Group in 2017. For each task, the start and end timestamps, required CPU resources, and data size were meticulously recorded. All such data types were categorized as type 0. Subsequently, we stored all the data information to construct the meta-service dataset. Upon completing the construction of both the virtual scene service dataset and the meta-service dataset, the two datasets were amalgamated into the metaverse service dataset, with variations in proportions and quantities.

TABLE II
VIRTUAL SCENE SERVICE DATASET

id	start_time	end_time	data_size(MB)	demand_cycle	task_type
1	2023/7/27 18:31	2023/7/27 18:32	626.1	401	1
2	2023/7/21 23:32	2023/7/21 23:32	763.3	427	1
3	2023/7/26 21:17	2023/7/26 21:18	810.4	515	1
...	...	...	...	...	1
50	2023/7/24 14:14	2023/7/24 14:14	792.6	573	1

TABLE III
META-SERVICE DATASET

id	start_time(s)	end_time(s)	data_size(MB)	demand_cycle	task_type
1	9121	9139	0.0081	50	0
2	11162	11192	0.0041	50	0
3	12120	12143	0.0054	50	0
...	...	...	...	...	0
5050	17537	17547	0.0107	100	0

B. Experimental Results

To address **RQ1**, we consider that in the metaverse, users often engage in multiple activities within a virtual scene. For instance, a user in a metaverse virtual fitness room may simultaneously request services such as running, listening to music, and online shopping. To capture this scenario, we selected a range of 505, 808, 1010, 2020, and 3030 metaverse services, with a proportion of 1:100 between virtual scene services and meta-services. This ratio reflects the realistic scenario in which each metaverse user can perform multiple activities in a virtual scene [21]. Fig. 3 presents the optimal average delay (s) for all services, virtual scene services, and meta-services. From the results, we can conclude that our approach satisfies the minimum delay requirements of mobile devices,

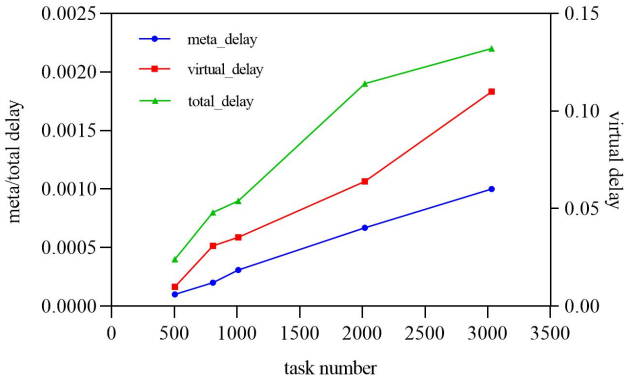


Fig. 3. The delay of various quantities of services

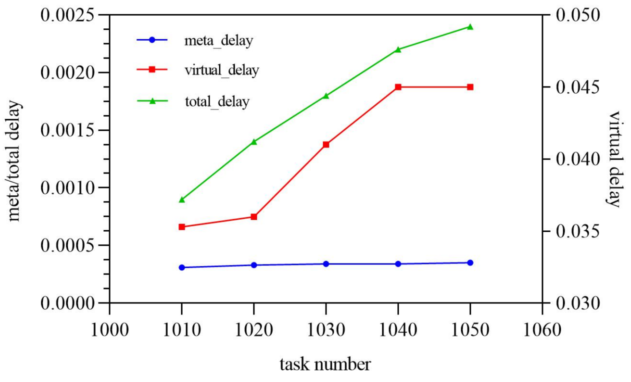


Fig. 4. The delay of different proportions of services

such as head-mounted displays, across varying quantities of metaverse services [29]. Additionally, we observe that the delay of individual virtual scene services has a more significant impact on overall delay than meta-services, reinforcing the significance of our research findings.

Furthermore, considering the realistic scenario where some users may prefer frequent switching between metaverse virtual scenes, we modified the ratio between virtual scene services and meta-services. Consequently, we selected sets of 1010, 1020, 1030, 1040, and 1050 services, with varying proportions of virtual scene services to metaverse services: 1:100, 1:50, 3:100, 1:25, and 1:20, respectively. Experimental results demonstrate the effectiveness of Meta-PPO across different ratios of virtual scene services to meta-services. Refer to Fig. 4 for specific details.

For **RQ2**, considering the significant dependence of heuristic algorithms on expert knowledge or accurate mathematical models, the process of updating models to tackle the proposed offloading problem can be time-consuming [30]. In contrast, reinforcement learning methods excel in capturing latent dynamics of an environment without any a priori knowledge, enabling the acquisition of optimal long-term goal strategies through iterative exploration within specific contextual settings [31]. Consequently, reinforcement learning is regarded as a well-suited approach for efficiently searching for optimal solutions in dynamic and evolving environments [32]. Hence, we selected three classic reinforcement learning algorithms,

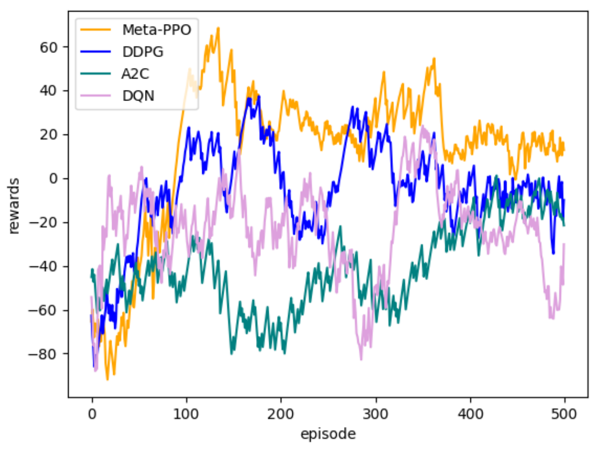


Fig. 5. Comparative analysis of reward values across different algorithms

namely DQN, DDPG, and A2C, to compare their performance with our proposed method, PPO. DQN was chosen as a representative of classic reinforcement learning algorithms, while DDPG was selected due to its suitability for handling multi-dimensional action problems [33]. A2C, although similar to PPO in some aspects [34], differs in terms of not performing replay of sampled data and gradient clipping. Therefore, we included DQN, DDPG, and A2C as the comparative algorithms. We conducted the comparison using a dataset consisting of a total of 1010 services, with a ratio of 1:10 between virtual scene services and meta-services. After 500 episodes of iterations, we evaluated the average sum of rewards and compared it with our proposed method. As depicted in Fig. 5, after 500 iterations, we observed the convergence of rewards for PPO, A2C, and DDPG, stabilizing at a range of approximately [0, 30]. A2C showed a similar convergence range to DDPG but with DDPG converging slightly earlier. In contrast, DQN did not exhibit a clear convergence trend even after 500 iterations. Based on these results, we can conclude that our proposed method outperforms the comparative algorithms. Fig. 6 and Fig.7 depict the average values of our optimization objectives under different algorithms. Fig. 6 illustrates the delay(s) for all services, virtual scene services, and meta-services in a dataset comprising 1010 instances and 500 episodes, with a ratio of 1:10 between virtual scene services and meta-services. Considering delay as a singular objective, our method outperforms DDPG and DQN significantly in terms of overall delay, virtual scene service delay, and meta-service delay. The optimization effect of A2C in terms of meta-service delay is similar to our proposed method. However, A2C exhibits noticeably higher virtual scene service delay compared to other methods. Such an optimization approach that sacrifices virtual scene service delay excessively is not suitable for users who frequently switch virtual scenes. The comparative effects of our optimization objective, energy consumption, are presented in Fig.7. Under the balanced strategy of energy consumption and memory, i.e., $\alpha = \beta = 0.5$, our method achieves a resource utilization rate of 59.5%. This represents

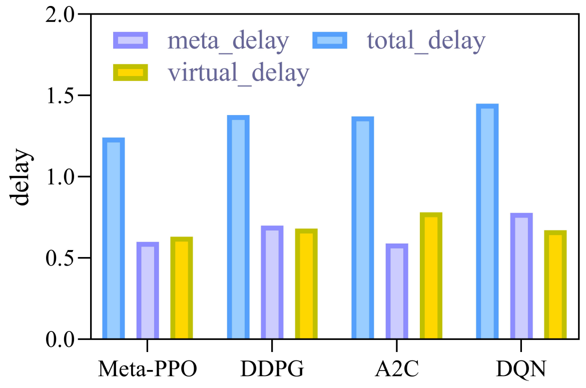


Fig. 6. The delay of different algorithms

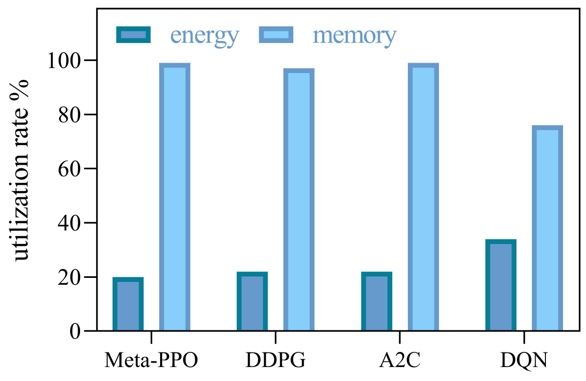


Fig. 7. The resource utilization rates of different algorithms

a 1% improvement over DDPG, a 9.5% improvement over DQN, and a 2% decrease to A2C. We attribute this to the fact that A2C excessively sacrifices delay, resulting in optimization strategies that prioritize resource utilization. Taking into account a comprehensive evaluation, our method exhibits superiority over alternative approaches.

To address **RQ3**, Considering the offloading of services to servers can effectively reduce computational delay but may lead to a decrease in resource utilization on mobile devices. we employed the concept of Pareto optimality to tackle this problem. Pareto optimality is a common solution in multi-objective optimization problems [35]. It is defined as a solution where no improvement in one objective can be made without degrading at least one other objective. Graphically, Pareto optimality is often depicted as a curve or boundary known as the Pareto Frontier. This frontier illustrates the trade-off relationships between various objectives, where each solution excels in some objectives over others but is relatively inferior in others.

In this paper, we sought the Pareto Frontier by employing different optimization algorithms such as DDPG, PPO, DQN, and A2C. Fig. 8 illustrates the Pareto Frontiers we discovered under these algorithms. From the graph, it can be observed that the Pareto points of the DDPG algorithm resemble the Pareto Frontier of the PPO algorithm. However, the Pareto points of the DDPG algorithm are excessively concentrated,

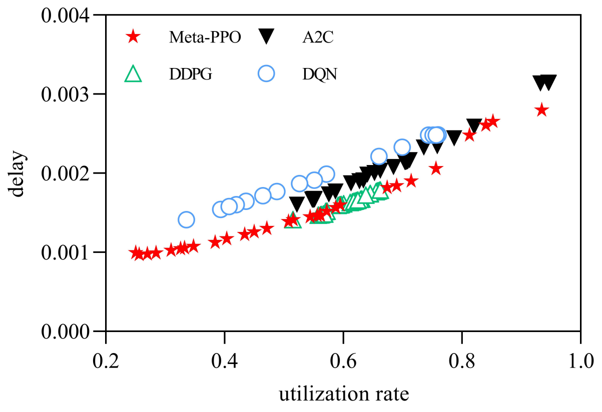


Fig. 8. Pareto frontier of different algorithms

indicating a limitation in balancing various objectives with this algorithm. On the other hand, the DQN and A2C algorithms each conducted relatively good exploration in the objectives of energy consumption and resource utilization, respectively. However, they were unable to explore these objectives more deeply simultaneously. In summary, our approach outperforms other optimization algorithms in approximating the Pareto Frontier. Our method excels in balancing various optimization objectives and achieves a better equilibrium between energy consumption and resource utilization, thereby obtaining superior solutions.

VI. Conclusion

In this paper, we investigate the metaverse and metaverse services, proposing a QoS optimization method applicable to the metaverse environment. We categorize metaverse services into virtual scene services and meta-services and make offloading decisions for these two types of services, aiming to minimize delay and maximize mobile device resource utilization under constraints of computation, bandwidth, energy consumption, and memory. Simulation results demonstrate the superiority of our proposed method over other traditional reinforcement learning methods.

For future work, we will further explore these issues: 1) We will investigate the computational offloading problem when multiple users compete for resources; 2) We will design solutions for fair resource allocation among multiple users and tasks; 3) We will delve deeper into the prioritization of services when combining a large number of virtual scene services and meta-services within metaverse services.

Acknowledgments

This work is funded by the National Natural Science Foundation of China under Grant No.U21B2016 and No.62272145, the Natural Science Research Startup Foundation of Recruiting Talents of Nanjing University of Posts and Telecommunications (Grant No. NY223166), and the Australian Research Council's Discovery Projects funding scheme (DP220101823).

References

[1] N. Stephenson, *Snow crash*. Penguin UK, 1994.

[2] A. Trujillo and C. Bacciu, "Toward blockchain-based fashion wearables in the metaverse: the case of decentraland," in *2023 IEEE International Conference on Metaverse Computing, Networking and Applications (MetaCom)*. IEEE, 2023, pp. 653–657.

[3] C. Dong, J. Zhou, Q. An, F. Jiang, S. Chen, and X. Liu, "Revolutionizing virtual shopping experiences: A blockchain-based metaverse uav delivery solution," in *2023 IEEE International Conference on Metaverse Computing, Networking and Applications (MetaCom)*. IEEE, 2023, pp. 658–661.

[4] T. Lan and J. Zhao, "Optimization in mobile augmented reality systems for the metaverse over wireless communications," *arXiv preprint arXiv:2311.17630*, 2023.

[5] J. Zhao, X. Zhou, Y. Li, and L. Qian, "Optimizing utility-energy efficiency for the metaverse over wireless networks under physical layer security," *arXiv preprint arXiv:2303.04683*, 2023.

[6] 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.

[7] T. Q. Duong, D. Van Huynh, S. R. Khosravirad, V. Sharma, O. A. Dobre, and H. Shin, "From digital twin to metaverse: The role of 6g ultra-reliable and low-latency communications with multi-tier computing," *IEEE Wireless Communications*, vol. 30, no. 3, pp. 140–146, 2023.

[8] Y.-J. Liu, H. Du, D. Niyato, G. Feng, J. Kang, and Z. Xiong, "Slicing4meta: An intelligent integration architecture with multi-dimensional network resources for metaverse-as-a-service in web 3.0," *IEEE Communications Magazine*, 2023.

[9] H. Zhang, S. Mao, D. Niyato, and Z. Han, "Location-dependent augmented reality services in wireless edge-enabled metaverse systems," *IEEE Open Journal of the Communications Society*, vol. 4, pp. 171–183, 2023.

[10] X. Huang, Y. Wu, J. Kang, J. Nie, W. Zhong, D. I. Kim, and S. Xie, "Service reservation and pricing for green metaverses: A stackelberg game approach," *IEEE Wireless Communications*, vol. 30, no. 5, pp. 86–94, 2023.

[11] K. Meng, Y. Hui, R. Sun, N. Cheng, Z. Su, and T. H. Luan, "Environment-aware dynamic resource allocation for vr video services in vehicle metaverse," in *2023 IEEE 98th Vehicular Technology Conference (VTC2023-Fall)*. IEEE, 2023, pp. 1–5.

[12] T. Zhao, S. Zhou, X. Guo, Y. Zhao, and Z. Niu, "A cooperative scheduling scheme of local cloud and internet cloud for delay-aware mobile cloud computing," in *2015 IEEE globecom workshops (GC Wkshps)*. IEEE, 2015, pp. 1–6.

[13] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, 2013.

[14] Y. Ge, Y. Zhang, Q. Qiu, and Y.-H. Lu, "A game theoretic resource allocation for overall energy minimization in mobile cloud computing system," in *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, 2012, pp. 279–284.

[15] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM transactions on networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[16] A. Haeberlen, L. T. X. Phan, and M. McGuire, "metaverse as a service: Megascale social 3d on the cloud," in *Proceedings of the 2023 ACM Symposium on Cloud Computing*, 2023, pp. 298–307.

[17] S. Xiao, C. Liu, K. Li, and K. Li, "System delay optimization for mobile edge computing," *Future Generation Computer Systems*, vol. 109, pp. 17–28, 2020.

[18] C. Shu, Z. Zhao, Y. Han, G. Min, and H. Duan, "Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 1678–1689, 2019.

[19] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2016.

[20] Y. Song, S. S. Yau, R. Yu, X. Zhang, and G. Xue, "An approach to qos-based task distribution in edge computing networks for iot applications," in *2017 IEEE international conference on edge computing (EDGE)*. IEEE, 2017, pp. 32–39.

[21] X. Xu, Q. Z. Sheng, B. Benatallah, Z. Chen, R. Gazda, A. E. Saddik, and M. P. Singh, "Metaverse services: The way of services towards the future," in *2023 IEEE International Conference on Web Services (ICWS)*. IEEE, 2023, pp. 179–185.

[22] X. Gao, W. Yi, Y. Liu, and L. Hanzo, "Multi-objective optimisation of urllc-based metaverse services," *IEEE Transactions on Communications*, 2023.

[23] Z. Huang and V. Friderikos, "Optimal mobility-aware wireless edge cloud support for the metaverse," *Future Internet*, vol. 15, no. 2, p. 47, 2023.

[24] 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, 2023.

[25] N. H. Chu, D. N. Nguyen, D. T. Hoang, K. T. Phan, E. Dutkiewicz, D. Niyato, and T. Shu, "Dynamic resource allocation for metaverse applications with deep reinforcement learning," in *2023 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2023, pp. 1–6.

[26] J. Liu, S. Guo, Q. Wang, C. Pan, and L. Yang, "Optimal multi-user offloading with resources allocation in mobile edge cloud computing," *Computer Networks*, vol. 221, p. 109522, 2023.

[27] J. Jeong, I.-M. Kim, and D. Hong, "Deep reinforcement learning-based task offloading decision in the time varying channel," in *2021 International Conference on Electronics, Information, and Communication (ICEIC)*. IEEE, 2021, pp. 1–4.

[28] A. Computing, "Reinforcement learning in autonomic computing."

[29] S. Friston, E. Griffith, D. Swapp, C. Lrondi, F. Jjunju, R. Ward, A. Marshall, and A. Steed, "Quality of service impact on edge physics simulations for vr," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 5, pp. 2691–2701, 2021.

[30] L. Zhao, E. Zhang, S. Wan, A. Hawbani, A. Y. Al-Dubai, G. Min, and A. Y. Zomaya, "Meson: A mobility-aware dependent task offloading scheme for urban vehicular edge computing," *IEEE Transactions on Mobile Computing*, 2023.

[31] Y. Li, X. Wang, X. Gan, H. Jin, L. Fu, and X. Wang, "Learning-aided computation offloading for trusted collaborative mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 12, pp. 2833–2849, 2019.

[32] W. Zhang, Z. Zhang, S. Zeadally, H.-C. Chao, and V. C. Leung, "Masm: A multiple-algorithm service model for energy-delay optimization in edge artificial intelligence," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4216–4224, 2019.

[33] X. Yang, H. Fang, Y. Gao, X. Wang, K. Wang, and Z. Liu, "Computation offloading and resource allocation based on p-dqn in leo satellite edge networks," *Sensors*, vol. 23, no. 24, p. 9885, 2023.

[34] S. Huang, A. Kanervisto, A. Raffin, W. Wang, S. Ontañón, and R. F. J. Dossa, "A2c is a special case of ppo," *arXiv preprint arXiv:2205.09123*, 2022.

[35] C. Zhu, L. Xu, and E. D. Goodman, "Generalization of pareto-optimality for many-objective evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 2, pp. 299–315, 2015.