Immersive Multimedia Service Caching in Edge Cloud with Renewable Energy

M. SHAMIM HOSSAIN, Department of Software Engineering, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia
YIXUE HAO, LONG HU, and JIA LIU, Huazhong University of Science and Technology, Wuhan, China
GANG WEI and CHEN MIN, South China University of Technology, Guangzhou, China

Immersive service caching, based on the intelligent edge cloud, can meet delay-sensitive service requirements. Although numerous service caching solutions for edge clouds have been designed, they have not been well explored. Moreover, to the best of our knowledge, there is no work to consider the immersive service caching scheme under the supply of renewable energy. In this article, we investigate the service caching problem under the renewable energy supply to minimize service latency while making full use of renewable energy. Specifically, we formulate the service caching and renewable energy harvesting problem, which considers the dynamic renewable energy, unknown service requests, and limited capacity of the edge cloud. To solve this problem, we propose an effective algorithm, called OSCRE. Our algorithm first uses Lyapunov optimization to convert the time-average problem into time-independence optimization and thus realizes optimal renewable energy harvesting. Then, it realizes the service caching scheme using data-driven combinatorial multi-armed bandit learning. The simulation results show that the OSCRE scheme can save service latency while making sufficient use of renewable energy.

CCS Concepts: • **Networks → Cloud computing;**

Additional Key Words and Phrases: Edge cloud, intelligent scheduling; renewable energy, multimedia service caching

ACM Reference Format:
M. Shamim Hossain, Yixue Hao, Long Hu, Jia Liu, Gang Wei, and Chen Min. 2024. Immersive Multimedia Service Caching in Edge Cloud with Renewable Energy. *ACM Trans. Multimedia Comput. Commun. Appl.* 20, 6, Article 173 (March 2024), 23 pages. https://doi.org/10.1145/3643818

This work was supported by the Researchers Supporting Project number (RSP2024R32), King Saud University, Riyadh, Saudi Arabia.

Authors' addresses: M. Shamim Hossain (Corresponding author), Department of Software Engineering, College of Computer and Information Sciences, King Saud University, Riyadh 12372, Saudi Arabia; e-mail: mshossain@ksu.edu.sa; Y. Hao (Corresponding author), L. Hu, and J. Liu, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, 430074, China; e-mails: yixuehao@hust.edu.cn, hulong@hust.edu.cn, jialiu0330@hust.edu.cn; G. Wei, School of Electronic and Information Engineering, South China University of Technology, 510640 China; e-mail: ecgwei@scut.edu.cn; M. Chen, School of Computer Science and Engineering, South China University of Technology, and Pazhou Laboratory, Guangzhou, 510640 China; e-mail: minchen@ieee.org.

ACM 1551-6857/2024/03-ART173
https://doi.org/10.1145/3643818

1 INTRODUCTION

Nowadays, with the development of novel technologies such as Augmented Reality, Virtual Reality, and Digital Twin, the Metaverse has attracted attention from the world by providing immersive multimedia experiences to mobile users [26]. However, with limited computing capacity, edge devices can hardly process such immersive services. Although cloud-based service processing can satisfy the computing demand of such services, it can hardly meet the latency demand. This is because the long network transmission required for the processing of such services at the cloud and the possible congestion of backbone networks. Fortunately, with the development of edge computing, edge cloud can narrow its distance from edge devices by deploying servers at the network edge [21]. Moreover, the Metaverse application can be rendered in advance on edge clouds to reduce the latency. Thus, it can meet the need of computing intensive and delay-sensitive service required by edge devices. However, in contrast with the strong capacity of the cloud in computing and storing, edge cloud has limited computing and storage capacity, which makes it unable to cache all the services required by the edge device. Thus, an important problem is to determine what kind of services are to be cached in the edge cloud, i.e., edge service caching problem [18].

Lots of works consider the service caching in edge cloud [3, 13, 17, 25]. When consider the limited communication, computing, and caching resources of edge cloud, many existing works propose the service caching strategy using traditional optimization. Furthermore, some works try to use the learning-based algorithm to design service caching scheme. For example, the authors in Reference [24] present the optimal service caching scheme with the **multi-armed bandit (MAB)** learning algorithm through learning the service request pattern. However, all the aforementioned service caching solutions are for the same purpose, that is, to minimize the latency of users in acquiring services. In practice, caching multimedia services on edge cloud consume the energy of such edge cloud and the massive deployment of edge cloud further brings huge energy consumption, which is obviously a new challenge for green computing.

To address this challenge, using renewable energy (e.g., solar energy) to supply the edge cloud is feasible scheme, which has attracted extensive attention in the academic and industrial world [12, 16, 27]. For example, Zhang et al. [27] have studied the communication performance of base stations and give the optimal scheme for energy harvesting under the supply of renewable energy. Xu et al. [23] investigated he task scheduling scheme under the renewable energy supply. Thus, importing green energy techniques into edge computation will not only realized diversified deployment of renewable energy, but also reduce the operating cost of edge servers.

However, none of the service caching schemes consider the edge cloud under the supply of renewable energy. Moreover, the dynamic arrival of renewable energy make the existing service caching scheme inapplicable. Thus, in this work, we consider the service caching scheme in edge cloud under the supply of renewable energy, as shown in Figure 1. The power supply to the edge cloud comes from renewable energy and grid, and the edge cloud needs to design the service caching strategy and energy harvesting decision. However, this caching scheme design exists the following challenge:

Computing-intensive and data-intensive tasks: Current Metaverse services, including three-dimensional (3D) entertainment games, demand substantial computing, communication, and storage resources for enabling real-time virtual-physical interactions, thereby delivering immersive experiences to mobile users. Nevertheless, the constrained computing resources of mobile devices present a significant challenge to these immersive multimedia applications [1, 9, 10, 22]. For instance, with a camera refresh rate of 60 Hz, the associated visual analysis and rendering must be completed within 16.7 ms to ensure real-time interaction [11].

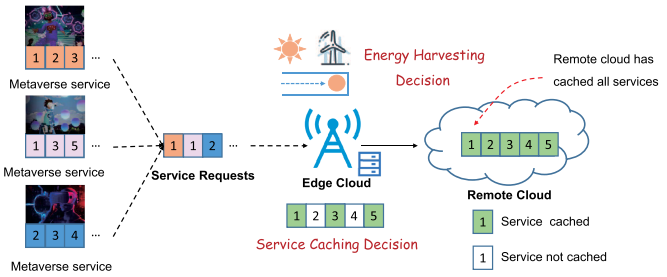

Fig. 1. An illustration of service caching in renewable energy enhanced edge computing framework.

Dynamic supply of renewable energy. Existing researches show that due to the influence of weather and other elements, the supply of renewable energy changes frequently, which brings new challenge to the systematic stability of the edge cloud [20]. For example, more services can be cached on the edge cloud when sufficient supply of renewable energy is available. Thus, we need design the service caching scheme and renewable energy harvesting scheme to adapt to the dynamic supply of renewable energy of the edge cloud.

Uncertain service demand. A significant challenge in edge cloud computing is the unpredictability of service demand. Many existing service caching schemes operate under the assumption that service demand is either known or follows a specific distribution, such as the Poisson distribution [25]. In contrast to this, our approach posits that service demand is inherently unpredictable, with edge devices potentially requesting services randomly. This unpredictability stems from the limited coverage of edge clouds and the high mobility of users. Additionally, considering safety and privacy concerns, it becomes challenging to access comprehensive user information.

To address these challenges, we propose the Online Service Caching with Renewable Energy (OSCRE) scheme, designed to optimize renewable energy supply and minimize immersive service request latency. By analyzing user service demands and the dynamics of renewable energy arrival, we formulate the service caching and energy harvesting problem, aiming to reduce service latency. Then, to solve this problem, we decompose the long-term renewable energy constraint with Lyapunov optimization. Our approach, distinct from existing methods, considers the impact of data size, computational load, and service storage capacity on service selection, with a focus on the data volume of immersive multimedia services. Consequently, we propose a Data-aware Combinatorial Upper Confidence Bound (D-CUCB) algorithm to manage unpredictable service requests, leading to an innovative online immersive service caching scheme. The effectiveness of OSCRE is validated using real-world MovieLens data [8]. This article's main contributions are summarized as follows:

— *Renewable energy enhanced edge service caching scheme*: We introduce a renewable energy-enhanced edge computing framework tailored for Metaverse services. By analyzing the supply of renewable energy alongside immersive service demands within the edge cloud, we formulate the service caching and energy harvesting problem, particularly under uncertain service demand scenarios. The objective of this optimization problem is to minimize service latency while adhering to the long-term constraints imposed by renewable energy.
— *OSCRE scheme*: We propose an effective online learning strategy, the OSCRE scheme, which facilitates online control of renewable energy and the online learning of Metaverse service request patterns. Specifically, OSCRE converts the long-term constraint of renewable energy into a per-time-slot optimization problem through Lyapunov optimization. By resolving each time slot issue, we achieve optimal energy harvesting control. Subsequently, we

develop an optimal service caching scheme using the D-CUCB algorithm. Ultimately, we demonstrate the OSCRE scheme's convergence.

— *Extensive performance evaluation*: We cconduct comprehensive experiments to assess the OSCRE scheme. These experiments demonstrate that, compared to various baseline algorithms, OSCRE effectively minimizes service latency and optimally utilizes renewable energy.

The remainder of this article is structured as follows: Section 2 reviews related work. Section 3 describes the system model and problem formulation. The OSCRE schemes are detailed in Section 4. Simulation results and discussions appear in Section 5, and Section 6 provides the conclusion.

2 RELATED WORKS

Our work is related two fields: (i) Metaverse service caching in edge cloud and (ii) service caching with renewable energy supply.

2.1 Metaverse Service Caching in Edge Cloud

With the development of the Metaverse, many immersive multimedia services have emerged, for example, online game, virtual shopping. In addition, traditional cloud-based services can hardly satisfy the user's latency demand. Fortunately, with the development of edge cloud, the edge cloud can satisfy the demand on the delay-sensitive services [21]. This is because by deploying servers at the network edge, the edge cloud can short distance between the edge cloud and the edge device. However,

Considering the limited capacity of the edge cloud in storing, computing, and communicating, a critical problem is what kind of services should be cached on the edge cloud. Here, service caching refers to cache the codes and running environment on the edge cloud. Researches have designed lots of service caching schemes [3, 7, 13, 17, 25]. Specifically, the service caching strategies can be divided into two categories, i.e., (i) service caching based on heuristic algorithm and (ii) learning-based service caching algorithm. For service caching based on heuristic algorithm, for example, the authors of Reference [17] studied the strategies for joint service caching and request scheduling in edge cloud and designed the quasi-optimal service caching strategy to minimize the service latency. Ma et al. [13] presented a service caching strategy based on the collaborative edge cloud. However, all these strategies are based on the assumption that the service demand is known a priori or follows specific distribution. Moreover, these algorithms generally need many times of iteration before reaching optimal convergency, which makes it difficult to guarantee the speed of convergency.

When considering that the service demand is unknown, the edge cloud needs to learn service request pattern of the edge device and online learning should be designed, such as MAB learning. Various works have studied the learning-based service caching scheme. MAB is an effective online learning algorithm that is capable to learn unknown information through balanced exploration (to learn unknown information) and exploitation (use learned knowledge) and finally make optimal decisions. This algorithm has been widely used in content caching, service caching, and networking. For example, the authors of Reference [4] propose a novel context-aware MAB algorithm that realizes the optimal caching strategies. Miao et al. [14] designed a service caching scheme using bandit learning in edge cloud and thus realized the maximization of the benefits of the service provider. Hao et al. [6] propose a service placement, workload scheduling and resource allocation strategies in multi-service edge cloud by using reinforcement learning. Although the service demand is unknown, the data volume preference can be learned by historical requests, the current works do not consider the influence of data volume of immersive multimedia service for the service caching when the service demand is unknown.

Considering the multidimensional constraints of computing, communication, and storage resources, Poularakis et al. [17] studied the optimization of joint service placement and request routing in dense edge computing networks, and the goal is to maximize the number of requests in the edge cloud. Xu et al. [25] studied the service caching problem in the mobile service market scenario. The optimization goal is to minimize the social cost of all network service providers, and propose integer linear programming and random rounding algorithm to solve the resource sharing problem between network service providers. To make full use of the storage and computing power of the edge cloud, Ma et al. [13] studied the service caching and workload scheduling problems in edge computing and developed an iterative algorithm to solve them. The service caching strategy is obtained based on Gibbs sampling, and then the workload scheduling problem is solved by a heuristic algorithm. However, caching services and processing tasks in edge clouds consume energy. When the energy of the edge cloud is insufficient, service caching and task processing may be interrupted. Existing edge service caching solutions rarely consider the issue of renewable energy. Therefore, it is crucial to address the problem of reducing energy consumption and operational costs in edge computing while ensuring service quality and promoting the sustainable development of the internet by fully utilizing renewable energy.

2.2 Service Caching with Renewable Energy Supply

All these works contribute a lot to service caching in edge cloud. However, most of the existing service caching strategies aim to minimize service latency. In practical, service caching and process consumes the energy of the edge cloud. Furthermore, according to the need of green computation, the energy problem of the edge cloud is another concern of the researches. Thus, using renewable energy to supply the edge system has become a feasible solution [12, 16, 20]. For example, the authors of Reference [15] discussed the solutions for task offloading of mobile devices under the supply of renewable energy. Zhang et al. [28] studied the task offloading strategies under the supply of renewable energy. Most of these researches focus on task offloading and resource management under the supply of renewable energy, and few works consider the service caching of edge cloud under renewable energy supply.

For the optimization of renewable energy, Lyapunov optimization is widely used. For example, Based on Lyapunov optimization, Guo et al. [5] consider an edge computing system with energy harvesting devices, and design an energy collection and computing offloading algorithm. To study the dynamic offloading and resource allocation in the multi-server edge cloud environment, Zhao et al. [29] collect renewable energy to minimize the system energy consumption and computing resources, and propose an online algorithm, which can schedule resources without predictive information.

Moreover, when the edge cloud is powered by renewable energy and such renewable energy is the main energy, it is necessary to consider simultaneously the energy consumption brought by service caching and the dynamic supply of renewable energy when designing the service caching strategies. An ideal solution is to cache services that have more user request when sufficient supply of renewable energy is available. However, the edge cloud system knows neither the size of supply of renewable energy nor the energy consumption for caching, in the next time slot, Thus, it is challenging to design the service caching under the supply of renewable energy.

3 SYSTEM MODEL AND PROBLEM DESCRIPTION

In this section, we introduce the immersive multimedia service caching framework in edge cloud with renewable energy, as shown in Figure 1, which consists of immersive multimedia applications, edge cloud, and remote cloud. The immersive multimedia applications connect to the edge cloud through wireless link and request services for edge cloud. The edge cloud processes the services

Table 1. The Summary Table of Importation Notations

Notation	Meaning
$\mathcal{N}$	The set of edge devices.
$\mathcal{K}$	The set of Metaverse services.
C	The maximum storage capacity of edge cloud.
μ_k	The computation amount required by service k.
c_k	The size of storage required by service k.
s_k^u	The size of task k.
x_k^t	Indicating whether the service k is cached on the edge cloud.
$d_{i,k}^t$	The number of request for service K by edge device i in time slot t.
f_e	The computing power of edge cloud.
f_c	The computing power of remote cloud.
γ	The wireless transmission rate.
ω	The backbone transmission rate.
τ	The round-trip time to the remote cloud.
E_C^t	The energy consumption of all services processed by edge cloud at time slot t.
E_H^t	The renewable energy obtained at time slot t.
B^t	The battery energy at time slot t.
G^t	Grid power purchased at time slot t.

requested by edge devices and we assume that the power supply to the edge cloud comes from both the renewable energy and the power grid, and the renewable energy is the main energy. Moreover, the remote cloud has all the services requested by the edge device.

3.1 System Overview

The Metaverse provides emerging applications including ESports, immersive social media, virtual training, and public or private digital places [11]. Thus applications require diverse immersive multimedia services, for example, high-fidelity images or videos for virtual scenes and 3D point cloud data for immersive interaction. These immersive multimedia services are time sensitive, requiring critically low latency and high accuracy. To provide close computing resources, we suppose that the edge cloud system has N of edge devices and is denoted by $\mathcal{N} = \{1, 2, \cdots, N\}$. Each device contains depth-sensing cameras (e.g., Microsoft Kinect) and Head Mounted Displays, and the camera can capture depth data, which is called point cloud data. The metaverse services library requested by the edge devices contains K immersive multimedia services, indexed by $\mathcal{K} = \{1, 2, \cdots, K\}$. Moreover, considering that different immersive multimedia services have different requirements for storage and computation, we use c_k to present the storage capacity required to cache service k, while w_k represents the computing resource required for processing service k. Furthermore, for ease of reference, we give the key notations in Table 1.

Although the edge cloud has storing and computing capacity to process the immersive multimedia service request by the edge devices, its resources in storing and computing are limited when compared to the remote cloud, which makes it unable to cache all the multimedia services. Thus, when the Metaverse service requested by the edge device is cached on the edge cloud, the edge

device can obtain this service from the edge cloud. Otherwise, the edge devices needs to get the service from the cloud. Therefore, we define a binary variable $x^t = (x_1^t, x_2^t, \cdots, x_K^t)$ at time slot t, and the $x_k^t \in \{0, 1\}$ denote the caching decision of service k at time slot t. When $x_k^t = 1$ means that service k is cached on the edge cloud at time slot t, and $x_k^t = 0$ means service k is not placed on the edge cloud at time slot t. Furthermore, considering the limited storage and computing power of edge cloud, we assume the storage capacity of edge cloud is C[bits]. Thus, the storage and computing capacity constraint can be expressed as $\sum_{k=1}^{K} x_k^t c_k \leq C$. Moreover, we divide time into discrete time slot $t = 1, 2, \cdots, T$, where T denotes the finite time horizon.

3.2 Energy Consumption Model

We first consider the energy consumption of service caching on the edge cloud. Specifically, the energy consumption of edge cloud includes the following three parts, i.e., (i) static energy consumption, (ii) immersive multimedia service processing energy consumption, and (iii) immersive multimedia service caching energy consumption. Since the energy consumption of service caching and processing is greater than the static energy consumption, thus, in this article, like Reference [14], we ignored the static energy consumption of the edge cloud.

Immersive Service processing energy consumption: According to Reference [3], it is related to the service processed and the CPU frequency assigned by the edge cloud. Assuming that the computing resource assigned by the edge cloud to the service k at time slot t is f_e, the processing energy consumption of service k at time slot t is as follows:

$$E_{k,t}^{comp}(x_k^t) = \nu \omega_k (f_e)^2 \sum_{i=1}^{N_t} x_k^t d_k^t, \tag{1}$$

where ν is unit energy consumption, and this parameter is relevant to the structure of edge cloud. The d_k^t is the number of request for service k at time slot t, and N_t is the number of edge devices covered by the edge cloud at time slot t.

Immersive service caching energy consumption: According to Reference [4], we assume that the unit energy consumption when edge cloud stores immervice service related data is η, for example, the 360° multimedia video content with the spherical feature. Thus, we can obtain the service caching energy consumption of k as follows:

$$E_{k,t}^{cach}(x_k^t) = \mathbf{1}(x_k^t = 1) s_k \eta, \tag{2}$$

where $\mathbf{1}\{\cdot\}$ is the indicator function, i.e., when the formula in brackets is true, the function value is 1. Based on the analysis above, we can obtain that the total energy consumption of the edge cloud at time slot t is as follows:

$$E_C^t(x^t) = \sum_{k=1}^{K} \left(E_{k,t}^{comp}(x_k^t) + E_{k,t}^{cach}(x_k^t) \right). \tag{3}$$

Renewable energy supply: We give the energy harvesting model of renewable energy. Importing renewable energy into edge computing can help realize green computation and reduce the energy consumption of the power grid. However, due to the weather and other reasons, the arrival of renewable energy is intermittent and unpredictable. Thus, we assume that the energy arriving at the edge cloud at time slot t is ε^t and that ε^t is independent identically distributed. We define E^{max} as the upper limit of renewable energy, in each time slot, part of the renewable energy reached will be collected and stored in battery, thus the renewable energy collected at time slot t (i.e., E_H^t) satisfied $0 \leq E_H^t \leq \varepsilon^t$.

However, when the supply of renewable energy is insufficient, the edge cloud needs to buy energy from the power grid to maintain the service caching and processing. We assume that the edge cloud purchase G^t of energy from the power grid in time slot t. In this work, consider the renewable energy is free, and, thus, the energy purchased from the power grid should be minimized as much as possible. Moreover, we assume that the renewable energy and the power grid energy are all stored in the battery, and define the battery capacity of the edge cloud at time slot t as B^t. Moreover, we assumed $B^0 = 0$, $B^t < +\infty$, and the maximum capacity of the battery is $B^{\max}$. Thus, we can obtain the dynamic change of battery capacity with the time as follows:

$$B^{t+1} = \min\{B^t - E_C^t(x^t) + E_H^t + G^t, B^{\max}\}. \tag{4}$$

Moreover, the energy consumption of the edge cloud should be smaller than the battery capacity at this time slot t, that is, $E_C^t(x^t) \leq B^t$. Based on the analysis above, we can see that the energy status of the battery changes continuously along the time. To guarantee the stability of the edge cloud system, according to Reference [29], we can change Equation (4) as the following long-term energy supply and consumption conditions:

$$\lim_{T\to+\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[E_c^t(x^t) - E_H^t - G^t] = 0. \tag{5}$$

3.3 Immersive Service Latency Model

The immersive service latency can be divided into communication latency and computation latency. The communication latency refers to the delay for the immersive service task transmission through wireless link or wired link. In view of the fact that the mobile device and the edge cloud is connected through wireless link, we define the wireless transmission rate is r_1. Furthermore, considering the wired link between the edge cloud and remote cloud, we define r_2 and τ as backbone transmission rate and the round-trip time, respectively. Therefore, we can obtain that when immersive service k is cached on the edge cloud (e.g., the real-time reasoning and high-fidelity rendering the multimedia in the edge cloud), its communication latency is c_k/r_1; otherwise, its latency is $c_k/r_1 + (c_k/r_2 + \tau)$.

The computation latency that refers to the latency of the immersive service being processed at edge cloud or remote cloud. We define f_e and f_c as the CPU frequency assigned to service k by the edge cloud and the remote cloud, respectively. Therefore, we can obtain the latency of service k being processed at the edge cloud and the remote cloud are ω_k/f_e and ω_k/f_c, respectively.

Based on the above analysis, we can obtain that when service k is cached on the edge cloud, the service latency as follows:

$$D^t(x^t) = \begin{cases} \sum_{k=1}^{K} d_k^t \left(\frac{c_k}{r_1} + \frac{\omega_k}{f_e}\right) & \text{if} \quad x_k^t = 1, \\ \sum_{k=1}^{K} d_k^t \left(\frac{c_k}{r_1} + \frac{c_k}{r_2} + \frac{\omega_k}{f_c} + \tau\right) & \text{if} \quad x_k^t = 0. \end{cases} \tag{6}$$

3.4 Problem Formulation

In this work, we focus on two problems: (1) Immersive service placement problem: What kind of services should be placed on the edge cloud? (2) Renewable energy harvesting problem: How much renewable energy should be obtained? However, when the edge cloud is powered by renewable energy, the design of service caching scheme is much more complicated that the traditional edge cloud system for power supply of the power grid. This is because in this condition, we need to process simultaneously the service demand information, the energy consumption information, and the service latency information. Furthermore, the battery capacity, which is associated with

the time, further complicates the service caching decision. Thus, the goal is to minimize the time-averaged latency of all service, which can be formulated as follows:

$$\mathcal{P}1: \min_{x^t, E_H^t} \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[D_t(x^t)] \tag{7}$$

$$s.t. \quad C1: \sum_{k=1}^{K} x^t c_k \le C.$$

$$C2: 0 \le E_H^t \le \varepsilon^t, t \in \mathcal{T}$$

$$C3: \lim_{T\to+\infty} \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[E_c(x^t)^t - E_H^t - G^t] = 0$$

$$C4: x^t \in \{0,1\}. t \in \mathcal{T}, k \in \mathcal{K},$$

where $C1$ guarantees that the services cached should not exceed the maximum storing capacity of the edge cloud. $C2$ indicates the limitation obtain of renewable energy, while $C3$ ensures the long term stability of energy supply and consumption of edge cloud system. $C4$ shows that whether the service is cached is a zero-one indicator.

However, for the solution of the above problem, following challenges exist:

— Since the battery power is dynamic, the current battery power is related to the previous power, the renewable energy supply and the energy consumption. Thus, the optimization problem $\mathcal{P}1$ results in the coupling of in time domain, i.e., the status at all the time slots is required to solve the problem. However, we can hardly know the status at all the time slots. To address this challenge, we utilize the Lyapunov optimization technology to convert the long-term constraints problem to a series of real-time minimization problem, and such optimization can be solved without knowing the information of future time slots. Thus, in this article, with Lyapunov optimization method, we realize the decoupling of optimization problem in time domain.

— For the number of immersive multimedia service demand d_k^t, most of existing works assume that the service demand of the edge device is known (e.g., follow specific distribution) or can be obtained through well-studied learning method (e.g., deep learning methods). However, when the service provider is making caching decisions at time slot t, it is very difficult to predict precisely the number of service demand at time slot t. Moreover, in a period, the service demand of the user may change along with the interest of such user. Thus, in this article, we will utilize the MAB to learn the number of demand of mobile device online.

4 ONLINE SERVICE CACHING UNDER RENEWABLE ENERGY SUPPLY

In this section, we give the detailed introduction of OSCRE algorithm. Specifically, we first utilize the Lyapunov optimization to decouple the renewable energy supply problem into a one-time-slot optimization problem. Then, we design a online algorithm for service caching.

4.1 Problem Transformation Based on Lyapunov Optimization

For the solution of the optimization problem $\mathcal{P}1$, a critical challenge is the randomness of the supply of renewable energy. To address this challenge, we utilize the disturbance Lyapunov optimization to solve the problem $\mathcal{P}1$, that is, to decouple the $\mathcal{P}1$ into an optimization problem that can be solved separately at each time slot. We first model the change in battery power as a queue. Then, we define the disturbance battery level of the edge cloud to be $\tilde{B}^t = B^t - \theta$, where θ is the

disturbance power parameter. Furthermore, we can define Lyapunov function as follows:

$$L(\tilde{B}^t) = \frac{1}{2}(\tilde{B}^t)^2 = \frac{1}{2}(B^t - \eta)^2. \tag{8}$$

Specifically, the Lyapunov function represents the "congestion level" in the energy queue. To ensure the stability of the energy deficit queue, we give the Lyapunov drift $\Delta(\tilde{B}^t)$ as follows:

$$\Delta(\tilde{B}^t) = \mathbb{E}[L(\tilde{B}(t+1)) - L(\tilde{B}^t)|\tilde{B}^t], \tag{9}$$

where $\Delta(\tilde{B}^t)$ is the one-time-slot expected changes about Lyapunov function, and the smaller $\Delta(\tilde{B}^t)$, the stabler battery queue. Bringing Equation (8) into Equation (9), we can obtain

$$\Delta(\tilde{B}^t) \leq B + \mathbb{E}[\tilde{B}^t(E_C^t(x^t) - E_H^t - G^t)], \tag{10}$$

where $B = \frac{1}{2}\left((E^{\max} + G^{\max})^2 + (E_C^{\max})^2\right)$, $E^{\max}$, and $E_C^{\max}$ are the upper bounds of E_H^t and $E_C^t(x^t)$, respectively, and $G^{\max}$ is the upper bound of the grid energy.

According to Lyapunov optimization theory, we define Lyapunov drift-plus-penalty as follows:

$$\Delta(\tilde{B}^t) + V\mathbb{E}[D^t(x^t)|\tilde{B}^t] \leq B + \mathbb{E}[VD^t(x^t) + \tilde{B}^t(E_H^t + G^t - E_C^t(x^t))|B^t], \tag{11}$$

where V is a constant parameter and used to adjust the tradeoff between the battery capacity and the service caching latency. Furthermore, by minimizing the upper bound of the Lyapunov drift-plus-penalty function, we can solve the optimal service caching strategy separately for each time slot. In other words, we can convert the original optimization problem $\mathcal{P}1$ into the following optimization problem $\mathcal{P}2$:

$$\begin{aligned} \mathcal{P}2: \min_{x^t, E_H^t} \; & VD^t(x^t) + \tilde{B}^t(E_H^t + G^t - E_C^t(x^t)) \\ s.t. \quad & C1, C2, C4. \end{aligned} \tag{12}$$

Thus, with the above analysis, we transform the long-term constraint of battery capacity into the per-time-slot problem. In each time slot, we need only to solve the optimization problem $\mathcal{P}2$ to get the optimal solution, and $\mathcal{P}2$ only needs the information of the current time as input. Details of the algorithm are as shown in Algorithm 1. Specifically, the line 2 of the algorithm first observe the battery capacity and the supply of renewable energy at the current time slot. In line 3, based on the information observed, solve the $\mathcal{P}2$ to obtain the optimal service caching and the collection of renewable energy. Finally, in line 4, based on the energy collection and the energy consumed, the algorithm updated the status of the battery capacity of the edge cloud and step to the next time slot.

ALGORITHM 1: The OSCRE Algorithm.

Input: N_t, c_k, ω_k, r_1, r_2, τ, and C
Output: Service caching scheme x^t and energy harvesting decision E_H^t.
1: **for** t from 0 to T-1 **do**
2: Observe the battery capacity B^t, renewable energy ε_t;
3: In time slot t, by solving the optimization problem $\mathcal{P}_2$ obtain the optimal energy harvesting E_H^t and service caching strategy x^t;
4: Update battery queue $B^{t+1} = B^t - E_C^t(x^t) + E_H^t + G^t$.
5: **end for**

Then, we give the solution of the optimization problem $\mathcal{P}_2$. We first analyze the objective function and can get the following function:

$$VD^t(x^t) - \tilde{B}^t(E_C^t(x^t) - E_H^t - G^t) = VD^t(x^t) - \tilde{B}^t E_C^t(x^t) + \tilde{B}^t(E_H^t + G^t). \tag{13}$$

From the above analysis, we can see that $VD^t(x^t) - \tilde{B}^t E_C^t(x^t)$ is an optimization problem associated with service caching and $\tilde{B}^t(E_H^t + G^t)$ is an optimization problem relevant to the renewable energy harvesting. Therefore, we can decompose the optimization problem $\mathcal{P}_2$ into two sub-problems, i.e., (i) energy harvesting problem to determine E_H^t and (ii) service caching problem to determine x^t.

We first consider the renewable energy harvesting, as shown in the following optimization problem $\mathcal{P}_3$:

$$\begin{aligned} \mathcal{P}_3 : \min_{E_H^t} & \tilde{B}^t(E_H^t + G^t) \\ s.t. \quad & C2. \end{aligned} \tag{14}$$

For the above optimization problem $\mathcal{P}_3$, using linear programming algorithm, we can obtain the optimal energy harvesting as $(E_H^t)^* = \varepsilon^t \mathbf{1}(\tilde{B}^t \leq 0)$. From the above formula, we can see that when $B^t < \theta$, all the renewable energy at the current time slot is accepted and the battery will be recharged. When $B^t > \theta$, there is no need to collect renewable energy. Moreover, when the collected renewable energy is insufficient, purchase the electricity required to meet the current energy consumption from the power grid. Next, we give the solution of service placement problem.

4.2 Data-aware Combinatorial UCB Online Service Caching Algorithm

According to the above discussion, the immersive multimedia service caching problem can be solved by the following optimization problem $\mathcal{P}_4$:

$$\begin{aligned} \mathcal{P}_4 : \min_{x^t} & \sum_{k=1}^{K} (VD_k^t(x^t) - \tilde{B}^t E_C^t(x^t)) \\ s.t. \quad & C1, C4. \end{aligned} \tag{15}$$

Since it is difficult to obtain the number of service requests in advance, in this work, we learn user requests through online learning.

4.2.1 Combinatorial MAB Model. The service caching problem can be formulated as a MAB problem, and each service can be regard as an arm and the service caching is is equivalent to the arm being selected. However, compared with the traditional MAB problem, the online service caching problem has the following two differences: (i) The edge cloud can cache multiple services at one time and need to meet storage resource constraints. Thus, this is a budgeted combinatorial MAB problem. (ii) We need to consider the impact of service attributes on the algorithm. Thus, we design a D-CUCB service caching algorithm, as shown in Figure 2. Specifically, first, we obtain the estimated service caching metric based on Lyapunov optimization. Then, we select the service according to the estimated service caching metric, and each time we select the service with the smallest metric for caching until it reaches the edge cloud storage capacity.

4.2.2 Lyapunov-based Immersive Service Caching Metric. To solve the optimization problem $\mathcal{P}_4$, we first define *Lyapunov-based immersive multimedia service caching metric* $\theta_k(t)$ as follows:

$$\theta_k(t) = VD_k^t(x_k^t) - \tilde{B}^t E_{C,k}^t(x_k^t). \tag{16}$$

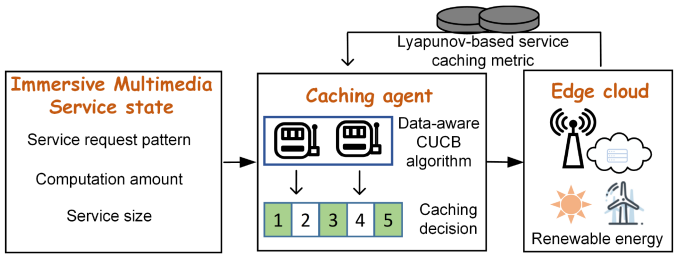

Fig. 2. Illustration of data-aware combinatorial UCB(D-CUCB) caching algorithm.

From Equation (16), we can see that whether the service is cached is associated with the service caching latency, the disturbance power of battery, and the service caching energy consumed. Furthermore, we define $n_k(t)$ to be the times of service k being chosen at time slot t as follows:

$$n_k(t) = \begin{cases} n_k(t-1)+1 & \text{if } k \in \mathcal{I}_t \\ n_k(t-1) & \text{if } k \notin \mathcal{I}_t \end{cases}, \tag{17}$$

where $\mathcal{I}_t$ is the set of the chosen service at time slot t. In the initialization, we set $n_k(0) = 0$.

Then, we denote $\bar{\theta}_k(t)$ as the edge cloud obtain average Lyapunov-based service caching metrics about service k up to time slot t, as follows:

$$\bar{\theta}_k(t) = \frac{\bar{\theta}_k(t-1)n_k(t-1) + \theta_k(t)}{n_k(t-1)+1}. \tag{18}$$

Moreover, we can utilize mean value $\bar{\theta}_k(t)$ as the estimated expected reward and the smaller the estimated expected reward, the more effective to reduce the delay. In addition to calculate the estimated expected reward, we also need to calculate the confidence. Based on the UCB algorithm, we utilize the $\sqrt{\frac{2\log(t)}{n_k(t-1)}}$ as confidence.

4.2.3 Service Caching Decision. We consider the relationship among the size of service, computation capacity required by the service, and the selection of services. For example, when the amount of computation required by a service is large, the service latency in the edge cloud processing is smaller than that in the local device, so it tends to cache the service. Similarly, when the size of the service is smaller, the delay of offloading the service to the edge cloud is smaller, and the cache effect is better. Thus, we define the data-aware parameter δ_k as follows:

$$\delta_k = l_1 \frac{1}{c_k} + l_2 \omega_k \tag{19}$$

where l_1 and l_2 is a weight parameter greater than 0. Specifically, we can obtain the *Lyapunov-based estimated service caching metric* (i.e., the criteria for the choice of service caching) of service k at time slot t as follows:

$$\hat{\theta}_k(t) = \bar{\theta}_k(t-1) - \rho\sqrt{\frac{2\delta_k \log(t)}{n_k(t-1)}}, \tag{20}$$

where ρ is a weighting parameter that represents the weight parameter of exploration and exploitation. Specifically, $\bar{\theta}_k(t-1)$ is the exploitation part, that is, the service whose caching brings smaller service latency will be cached. The second part is the exploration part, that is, the service with less times of exploration will be cached. Thus, the algorithm realizes the balance between exploration

and exploitation. Moreover, we denote the service chosen at time slot t to be k_t^*, as follows:

$$k_t^* = \arg\min_{k \in \mathcal{K}} \hat{\theta}_k(t). \tag{21}$$

ALGORITHM 2: D-CUCB Service Caching Algorithm

Input: $\tilde{B^t}$, V, K and C
Output: Service caching decision at time slot t: x_k^t
1: In each time slot t
2: **if** Any service $k \in \mathcal{K}$ has not been selected **then**
3: select service k to cache in the edge cloud
4: update $n_k(t) = n_k(t-1) + 1$
5: update the average Lyapunov-based service caching metric $\bar{\theta}_k(t) = \frac{\bar{\theta}_k(t-1)n_k(t-1)+\theta_k(t)}{n_k(t-1)+1}$
6: **else**
7: Calculate the Lyapunov-base estimated service caching metrics of each candidate services $k \in \mathcal{K}$ $\hat{\theta}_k(t) = \bar{\theta}_k(t) - \rho\sqrt{\frac{2\delta_k \log(t)}{n_k(t-1)}}$, and set $\mathcal{I}_t = \emptyset$
8: **while** $\sum_{k_t^* \in \mathcal{I}_t} c_{k_t^*} \leq C$ **do**
9: Select the service k according to the minimum Lyapunov-base estimated service caching metric $k_t^* = \arg\min_{k \in (\mathcal{K} \setminus \mathcal{I}_t)} \hat{\theta}_k(t)$
10: add service k_t^* into $\mathcal{I}_t$, i.e., $\mathcal{I}_t = \mathcal{I}_t + \{k_t^*\}$
11: update $n_k(t) = n_k(t-1) + 1$
12: update $\bar{\theta}_k(t) = \frac{\bar{\theta}_k(t-1)n_k(t-1)+\theta_k(t)}{n_k(t-1)+1}$
13: **end while**
14: **end if**

Moreover, the edge cloud can cache multiple services each time slot. In this work, we use the greedy algorithm to select, i.e., according to Lyapunov-based estimated service caching metric, each time slot will select services in a greedy way until reaching the edge clouid storing capacity. Specifically, the service caching algorithm is as shown in Algorithm 2. From line 2 to 6, the algorithm is in the initialization stage, and each service is cached once to obtain the Lyapunov-base estimated service caching metrics and the number of times selected. In line 7, the algorithm calculates the Lyapunov-base estimated service caching metrics of each candidate services. From line 8 to 10, the algorithm constantly select the service with the minimum Lyapunov-based estimated service caching metric to add the caching subset until reach the edge cloud caching capacity. From line 11 to line 12, the algorithm update the corresponding average Lyapunov-base service caching metrics and the number of times selected for each service accordingly.

4.3 Theoretic Analysis

Theorem 4.1. *The OSCRE algorithm can achieves the following bound on the time-averaged service latency:*

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[D_t(x^t)] \geq \frac{B}{V} + D^*, \tag{22}$$

and we can also obtain that the battery queue of the edge cloud tends to be stable, that is, there is $\varepsilon > 0$, *which makes Equation* (23) *is true,*

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[\tilde{B^t}] \geq \frac{B}{\delta} + \frac{V(D^{\max} - D^*)}{\delta}, \tag{23}$$

where D^* is the optimal value of $\mathcal{P}_2$, $D^{\max}$ is the maximum service lantency, and δ is a constant. Based on the above theorem, we can obtain that when the weight factor V increases, the solution of $\mathcal{P}_2$ is infinitely close to $\mathcal{P}_1$. However, when V increases, the battery queue also grow, and the system stability is affected. Thus, a reasonable V can balance the delay and queue length. Thus, we need choose a good V to achieve the tradeoff between delay and queue length. The detailed proof is give in the Appendix A.1.

Now we give the regret analysis of the D-CUCB algorithm. Since the number of user requests is independent, we assume that the service caching metric is also independent and identically distributed. We define the expectation of $\theta_k(t)$ as $\mathbb{E}[\theta_k(t)] = \mu_k$ and define the optimal service caching policy as $\mu^* = \min_{k \in \mathcal{K}} \mu_k(t)$. Thus, we can obtain the expected learning regret R_t (i.e., the gap between the currently selected cache service expected metric and the optimal cache expected metric) at time slot t as follows:

$$E(R_t) = \sum_{k \in \mathcal{I}_t} E(n_k(t))(\theta_k(t) - \mu^*). \tag{24}$$

Based on this, the expected cumulative learning regret R_T of the algorithm as $R_T = \sum_{t=1}^{T} E(R_t)$.

Theorem 4.2. *Under D-CUCB algorithm, the upper bound of the $\mathbb{E}[R_t]$ is as follows:*

$$\mathbb{E}[R_t] \leq \sum_{k=1}^{K} \frac{8\rho^2 \delta_k \log t}{\Delta_k} + O(1), \tag{25}$$

where $\Delta_k = \theta_k(t) - \mu^*$; Appendix 2 shows the detailed proof.

5 PERFORMANCE ANALYSIS

In this section, we evaluate the performance of OSCRE scheme. First, we introduce the experimental environment and the specific parameter settings. Then, we analyze the performance of the algorithm and introduce several comparison algorithms as the baseline. Finally, we discuss and evaluate the impact of different parameters on the performance of the algorithm.

5.1 Experiment Setup

In this experiment, we consider the renewable energy enhanced edge computing framework that includes edge cloud and metaverse service users. The edge cloud is deployed near to the wireless access network and the mobile devices are connected to the edge cloud through wireless channel. Consider a base station with an edge cloud server, and the transmission rate of the wireless network r_1 is 100 Mbps, and the transmission rate of the core network r_2 is 1 Gbps. For the computing power, we set the maximum computing power of the edge cloud f_e is 10 GHz, and the maximum computing power of the remote cloud f_c is 50 GHz. Moreover, the round-trip time is 500 ms. For the immersive multimedia services, we use the real-world MovieLens dataset [8], which includes 20 million ratings applied to 27,000 movies by 138,000 users to validate the efficiency of the A3C-based mobile edge caching policy. We assume that every movie rating is an immersive service request, since the rating data exhibits similar access patterns to those of content request data [19]. More specifically, we assume that every movie comment in our dataset is a downloading/streaming request, and the time when the comment is posted is considered the time when the request is initiated. Moreover, we set the number of time slots in the experiment to 300, and for each time slot run 100 experiments and take the average value. The specific parameters are shown in Table 2.

Table 2. Main Parameter Setting of the Experiment

Notation	meaning
the wireless transmission rate, r	100 Mbps
the backbone transmission rate, ω	1 Gbps
computing power of edge cloud, f_e	10 GHz
the storage capacity of edge cloud, C	500 GB
computing power of cloud, f_c	50 GHz
computation amount required by service k, μ_k	[0.1,0.5] GHz/task
the size of service k, c_k	[20,100] GB
the number of services, $\mathcal{K}$	20
the round-trip time to remote cloud, τ	0.5 s
Lyapunov parameter, V	0.5

5.2 Performance Analysis

Through Lyapunov optimization (for simplicity, write as Lya), the original long-time optimization problem (i.e., $\mathcal{P}1$) is transformed into a problem that can be solved separately at each time. After obtaining the optimal energy harvesting scheme, different algorithms can be used to solve the cache decision. To verify the performance of the OSCRE algorithm, for service caching decision, we compared with the following cache algorithms:

— *Lya with Random algorithm:* This randomly selects some services to cache on the edge cloud in each time slot, until reach the edge cloud storage capacity.
— *Lya with Softmax algorithm:* This is an improvement of ε-greedy cache algorithm. The ε-greedy cache algorithm exploits the optimal service with the probability of $1-\varepsilon$ and explores with the probability of ε. The Lya with Softmax greedy algorithm selects according to the rules of Softmax, so that some obviously poor service can be excluded, which reduces unnecessary exploration.
— *Lya with UCB algorithm:* This randomly selects a group of services within the edge cloud storage capacity and selects the optimal arm to replace one of them through traditional UCB algorithm.
— *Lya with CUCB algorithm:* After Lyapunov optimization, we adopt the CUCB algorithm for the service caching decision. The CUCB algorithm is an improvement of the UCB algorithm, i.e., in each time slot, a subset of arms with large reward value will be selected, but compared with the OSCRE algorithm, the data-awareness of service is not considered.

We utilize the learning regret to measure the convergence of the algorithm. The learning regret indicates the gap between the currently selected cache service metric and the optimal cache metric, i.e., the expected loss caused by not choosing the best service. We define the learning regret of the algorithm as $\sum_{k\in\mathcal{L}_t}\mathbb{E}[\theta_k(t)]-\mathbb{E}[\theta_k(t)^*]$, where $*$ is the optimal service and $\mathcal{L}_t$ is the set of service selected in time slot t. Based on this, we define the cumulative learning regret as

$$\sum_{\tilde{\tau}=1}^{t}\sum_{k\in\mathcal{L}_{\tilde{\tau}}}\mathbb{E}[\theta_k(\tilde{\tau})]-\mathbb{E}[\theta_k(\tilde{\tau})^*]. \tag{26}$$

Figure 3 shows the change curve of learning regret of five algorithms. We observe that when $t<20$, it is the initial stage of the algorithms, that is, each immersive multimedia service is cached once to obtain the reward, and when $t>100$, the learning regret slowly tends to stabilize. In

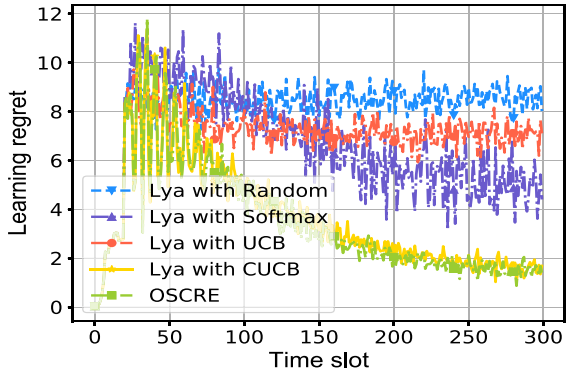

Fig. 3. Learning regret.

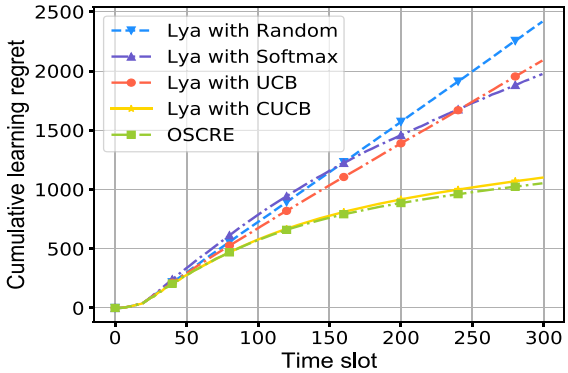

Fig. 4. Cumulative learning regret.

Figure 4, we know that the cumulative learning regret of the five algorithms shows a linear growth trend, which means that the learning regret of each time slot changes little with time. Moreover, we can see that, compared with other algorithms, when the algorithms converges, the curve fluctuation of OSCRE algorithm is small, and the learning regret is the lowest. The Lya with the Random algorithm randomly selects the services to be cached in each time slot, regardless of the impact of service requests and other factors, so the learning regret and cumulative learning regret are the highest. The Lya with the UCB algorithm only selects one optimal service in each time slot, so its performance is general. The Lya with Softmax has a certain exploration rate, so the learning regret value of the algorithm fluctuates greatly. The Lya with CUCB has relatively fast and stable convergence speed but does not consider the influence of the service, and therefore it is not optimal in service caching scenario. Thus, the OSCRE algorithm is better than caching benchmark algorithms in convergence.

Furthermore, we evaluate the Lyapunov-based average service caching metric under different algorithms, as shown in Figure 5. From the figure, we can see that, in the initial stage, the average service caching metric of the five algorithms fluctuate for a short time, then gradually decrease and tend to be stable. Among the five algorithms, the average service caching metric of ORCRE algorithm is the smallest, which shows that the optimization effect of ORCER algorithm is better than these baseline algorithms.

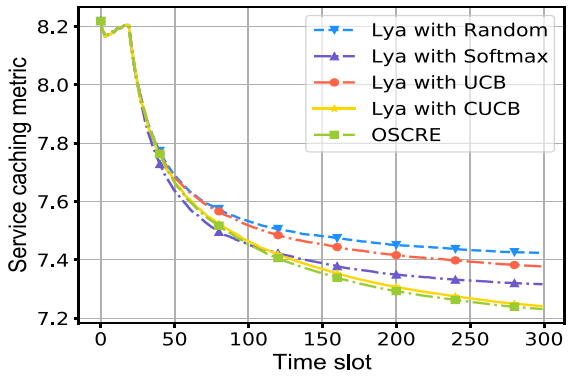

Fig. 5. Lyapunov-based immersive service caching metrics.

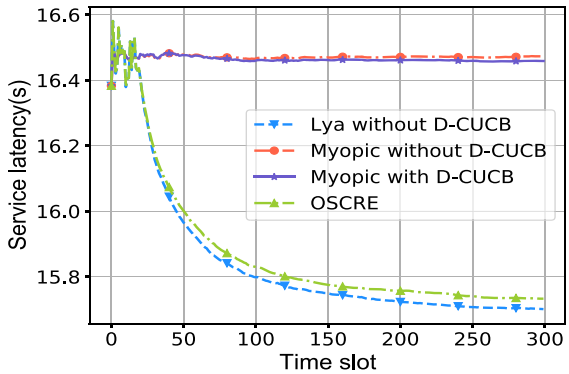

Fig. 6. Time average delay of five algorithms.

Then, to further evaluate the overall performance of the OSCRE algorithm, we compared the OSCRE with the following three benchmarks:

— *Myopic with D-CUCB:* It utilizes the D-CUCB algorithm to select the service with small latency to cache in the edge cloud, without considering the energy collection in the future.
— *Lya without D-CUCB:* In this algorithm, according to the supply of renewable energy, the edge cloud uses Lyapunov algorithm and UCB to cache the service instead of D-CUCB algorithm.
— *Myopic without D-CUCB:* In this algorithm, neither the energy collection at the future time nor the service caching scheme using D-CUCB algorithm are considered, and the service is cached randomly only when the edge cloud capacity constraint is satisfied.

Since Myopic with D-CUCB and Myopic without D-CUCB algorithms do not use Lyapunov optimization to transform the problem, and the service caching metric of the algorithm based on Lyapunov optimization can no longer be used to evaluate the algorithm performance, and, thus, we use the service latency under the condition of meeting the long-term energy consumption constraint.

As shown in Figure 6, when the time slot $t < 50$, the algorithm is in the initialization stage, and the four algorithm curves fluctuate to varying degrees. With the increase of time, it gradually tends to be stable. Moreover, we can see that compared with other algorithms, the OSCRE algorithm has

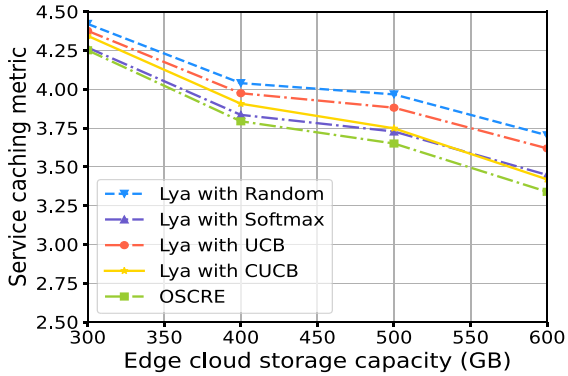

Fig. 7. The impact of edge cloud storage capacity on the Lyapunov-based service caching metric.

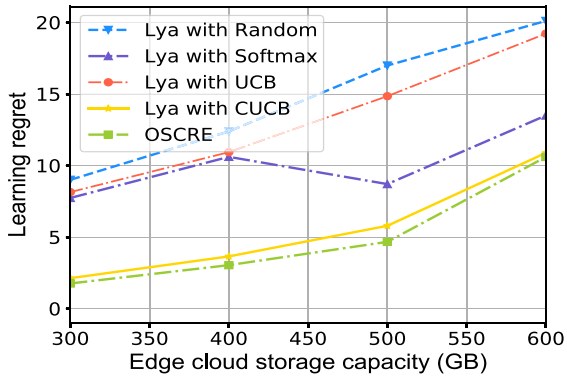

Fig. 8. The impact of edge cloud storage capacity on the learning regret.

the least service latency, and this shows that OSCRE algorithm has greatly improved the system performance.

5.3 Impact of Parameters on System Performance

Impact of edge cloud storage capacity on OSCRE: In this experiment, we set the storage capacity required for the service to be in the range of $[20, 100]$ GB, and the change range of edge cloud capacity is 300GB to 600 GB. As shown in Figure 7, with the increase of edge cloud storage capacity, the Lyapunov-based estimated service caching metric of the algorithms becomes smaller. This is because as the storage capacity of the edge cloud increases, the edge cloud can cache more services, which greatly saves the transmission delay and computing energy consumption. Thus, the Lyapunov-based estimated service caching metric is reduced.

Furthermore, Figure 8 describes the impact of edge cloud storage capacity on the learning regret of the algorithms. We observe that with the storage capacity of the edge cloud increases, the learning regret of the algorithms also increase. This can be explained that the edge cloud can cache more services, and the learning regret of algorithms is related to the number of cached services. Thus, increasing the edge cloud storage capacity can improve the performance of the algorithm. Moreover, from Figure 7 and Figure 8, we can also conclude that, compared with the other four algorithms, the ORCRE algorithm is superior to other algorithms under different edge cloud storage capacity.

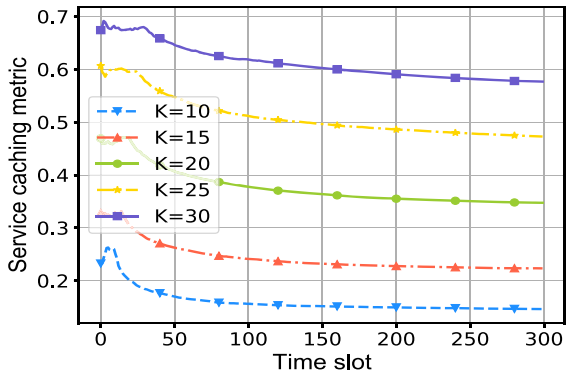

Fig. 9. The impact of number of services on the Lyapunov-based service caching metric.

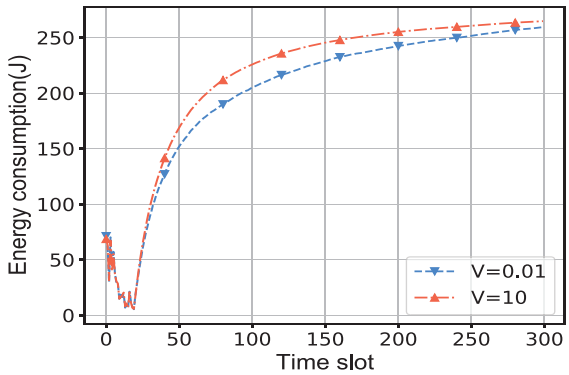

Fig. 10. The impact of parameter V on system average energy consumption.

Impact of services number on OSCRE: In this experiment, we set the number of service K from 10 to 30 for simulating the number of services from less to more. Figure 9 shows that with the increase of the number of services, the Lyapunov-based estimated service caching metric increases gradually. This is because when the number of services increases, more services will be processed in the edge cloud or remote cloud, and the delay and energy consumption of the system will also increase. For example, when the number of services $k = 30$, the Lyapunov-based estimated service caching metric of OSCRE is the largest. Moreover, when the number of services increases by 20, the service caching metric of the algorithm increases by 79.31%; this indicate that the number of services has a great impact on the system performance. Thus, with the number of service increases, if we ensure the performance of the algorithm remains unchanged, then we need to increase the storage capacity and computing resources of the edge cloud.

Impact of different V on OSCRE: We analyze the impact of the V in Lyapunov optimization on the system energy consumption and service latency, as shown in Figure 10 and Figure 11. From Figure 10, we can see that the higher the value of V, the greater the energy consumption. This can be explained as the larger the V value, the more the system pays attention to the latency. Figure 11 shows the influence of different V on service delay, and we can see that that the larger the value of V, the smaller the system delay. Moreover, from the above discussion, we can conclude that the tradeoff between system delay and energy consumption queue can be achieved by adjusting the

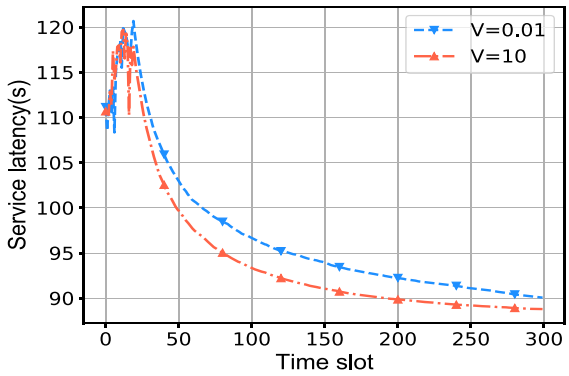

Fig. 11. The impact of parameter V on service average latency.

value of V, and when the battery energy is sufficient, we can reduce the value of V to achieve a smaller system delay, which is consistent with the conclusion of Theorem 1.

6 CONCLUSION

In this article, based on the analysis of renewable energy and service request patterns, we formulate the service caching and renewable energy harvesting problem under unknown service demand. To solve this problem, we propose the OSCRE algorithm using Lyapunov optimization and online learning. Finally, we evaluate the OSCRE algorithm, and our studies show that, when compared to other algorithms, the OSCRE algorithm can effectively minimize service latency while meeting long-term energy consumption constraints.

APPENDIX

A RESEARCH METHODS

A.1 Proof of Theorem 1:

First, we give the proof of Equation (22). To prove it, we first give the following lemma. According to Theorem in Reference [29], we can obtain that, for any $\varphi > 0$, $\mathcal{P}_2$ exists a stationary random strategy π, which can make $E_H^{\pi,t}$ and $x^{\pi,t}$ satisfy the following inequality:

$$\mathbb{E}[D^t(x^{\pi,t})] \leq D^* + \varphi \quad \mathbb{E}[E_H^{\pi,t} + G^{\pi,t} - E_C^t(x^{\pi,t})] \leq \varphi. \tag{27}$$

Since the algorithm can obtains the optimal solution of each time slot, we can choose among all possible decisions that can make $\mathcal{P}_2$ minimization strategy, including the above strategy π. Thus, we bring Equation (27) into Equation (11), and considering that the strategy π does not affect the battery energy queue, we can get the following equation:

$$\begin{aligned} &\Delta(\tilde{B}^t) + V\mathbb{E}[D^t(x^{\pi,t})|\tilde{B}^t] \\ &\leq B + \mathbb{E}[VD^t(x^{\pi,t}) + \tilde{B}^t(E_H^t + G^{\pi,t} - E_C^t(x^{\pi,t}))|B^t] \\ &\leq B + V(D^* + \varphi) + \varphi\tilde{B}^t. \end{aligned} \tag{28}$$

Making $\varphi \to 0$, we can obtain that

$$\Delta(\tilde{B}^t) + V\mathbb{E}[D^t(x^{\pi,t})|\tilde{B}^t] \leq B + VD^*. \tag{29}$$

Taking the expectation on both sides of Equation (29), summing within the range of $t \in \{0, 1, 2, \cdots, t-1\}$, and dividing both sides of the result by T, we can obtain the following

formula:

$$\frac{V}{T}\sum_{t=0}^{T-1}\mathbb{E}[D^t(x^{\pi,t})] \le B + VD^* - \frac{1}{T}\mathbb{E}[L(\tilde{B}^t) - L(\tilde{B}^0)]. \tag{30}$$

Since $L(\tilde{B}^0) = 0$, $L(\tilde{B}^t) \ge 0$, we make $T \to \infty$ and divide both sides of Equation (30) by V, and Equation (22) is proved.

Next, we analyze the stability of the battery queue; in each time slot, there exists a $\delta > 0$ and $\phi(\delta)$, and a policy $E_H^{\tau,t}$ and $x^{\tau,t}$, and we can satisfy the following equation:

$$\mathbb{E}[D^t(x^{\tau,t})] = \phi(\delta) \quad \mathbb{E}[E_H^{\tau,t} + G^{\tau,t} - E_C^t(x^{\tau,t})] \le -\delta. \tag{31}$$

Bringing Equation (31) into Equation (11), we can obtain that

$$\Delta(\tilde{B}^t) + V\mathbb{E}[D^t(x^{\tau,t})|\tilde{B}^t] \le B + V\phi(\delta) - \Delta\tilde{B}^t. \tag{32}$$

Taking the expectation on both sides of Equation (32), summing within the range of $t \in \{0, 1, 2, \cdots, t-1\}$, and dividing both sides of the result by T, we can get the following formula:

$$\begin{aligned} &\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\tilde{B}^t] \\ &\le \frac{B + V\left(\phi(\delta) - \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[D^t(x^{\tau,t})] - \frac{1}{T}\mathbb{E}[L(\tilde{B}^t) - L(\tilde{B}^0)]\right)}{\delta} \\ &\le \frac{B}{\delta} + \frac{V(D^{\max} - D^*)}{\delta}. \end{aligned} \tag{33}$$

Let the above formula $t \to \infty$, we can prove Equation (31).

A.2 Proof of Theorem 2:

According to the Theorem 1 in Reference [2], we can obtain that the upper bound of the expected value of the number of times the service is selected as follows:

$$\mathbb{E}[n_k(t)] \le \frac{8\rho^2\delta_k \log t}{\Delta_k^2} + M, \tag{34}$$

where M is a constant. Then, bringing Equation (34) into the regret value of algorithm, we can obtain

$$\mathbb{E}[R_t] = \sum_{k=1}^{K}\mathbb{E}[n_k(t)]\Delta_k \le \sum_{k}^{K}\left(\frac{8\rho^2\delta \log t}{\Delta_k} + M\right) \le \frac{8\rho^2\delta_k \log t}{\Delta_k^2} + M. \tag{35}$$

From Equation (35), we can conclude that the regret of the ORCRE algorithm has upper bound. Thus, the ORCRE algorithm is convergence.

REFERENCES

[1] Kazi Masudul Alam, Abu Saleh Md Mahfujur Rahman, and Abdulmotaleb El Saddik. 2013. Mobile haptic e-book system to support 3D immersive reading in ubiquitous environments. *ACM Trans. Multimedia Comput. Commun. Appl.* 9, 4, Article 27 (Aug. 2013), 20 pages.

[2] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.* 47, 2 (2002), 235–256.

[3] Lixing Chen, Jie Xu, Shaolei Ren, and Pan Zhou. 2018. Spatio–temporal edge service placement: A bandit learning approach. *IEEE Trans. Wireless Commun.* 17, 12 (2018), 8388–8401.

[4] Penglin Dai, Zihua Hang, Kai Liu, Xiao Wu, Huanlai Xing, Zhaofei Yu, and Victor Chung Sing Lee. 2020. Multi-armed bandit learning for computation-intensive services in MEC-empowered vehicular networks. *IEEE Trans. Vehic. Technol.* 69, 7 (2020), 7821–7834.

[5] Mian Guo, Qirui Li, Zhiping Peng, Xiushan Liu, and Delong Cui. 2022. Energy harvesting computation offloading game towards minimizing delay for mobile edge computing. *Comput. Netw.* 204 (2022), 108678.
[6] Yixue Hao, Min Chen, Hamid Gharavi, Yin Zhang, and Kai Hwang. 2020. Deep reinforcement learning for edge service placement in softwarized industrial cyber-physical system. *IEEE Trans. Industr. Inf.* 17, 8 (2020), 5552–5561.
[7] Yixue Hao, Min Chen, Long Hu, M. Shamim Hossain, and Ahmed Ghoneim. 2018. Energy efficient task caching and offloading for mobile edge computing. *IEEE Access* (2018), 11365–11373.
[8] F Maxwell Harper and Joseph A. Konstan. 2015. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.* 5, 4 (2015), 1–19.
[9] Yuna Jiang, Jiawen Kang, Dusit Niyato, Xiaohu Ge, Zehui Xiong, Chunyan Miao, and Xuemin Shen. 2023. Reliable distributed computing for metaverse: A hierarchical game-theoretic approach. *IEEE Trans. Vehic. Technol.* 72, 1 (2023), 1084–1100. https://doi.org/10.1109/TVT.2022.3204839
[10] Conor Keighrey, Ronan Flynn, Siobhan Murray, and Niall Murray. 2021. A physiology-based QoE comparison of interactive augmented reality, virtual reality and tablet-based applications. *IEEE Trans. Multimedia* 23 (2021), 333–341.
[11] Uman Khalid, Muhammad Shohibul Ulum, Ahmad Farooq, Trung Q. Duong, Octavia A. Dobre, and Hyundong Shin. 2023. Quantum semantic communications for metaverse: principles and challenges. *IEEE Wireless Commun.* 30, 4 (2023), 26–36. https://doi.org/10.1109/MWC.002.2200613
[12] Meng-Lin Ku, Wei Li, Yan Chen, and K. J. Ray Liu. 2015. Advances in energy harvesting communications: Past, present, and future challenges. *IEEE Commun. Surv. Tutor.* 18, 2 (2015), 1384–1412.
[13] Xiao Ma, Ao Zhou, Shan Zhang, and Shangguang Wang. 2020. Cooperative service caching and workload scheduling in mobile edge computing. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'20)*. IEEE, 2076–2085.
[14] Yiming Miao, Yixue Hao, Min Chen, Hamid Gharavi, and Kai Hwang. 2020. Intelligent task caching in edge cloud via bandit learning. *IEEE Trans. Netw. Sci. Eng.* 8, 1 (2020), 625–637.
[15] Minghui Min, Liang Xiao, Ye Chen, Peng Cheng, Di Wu, and Weihua Zhuang. 2019. Learning-based computation offloading for IoT devices with energy harvesting. *IEEE Trans. Vehic. Technol.* 68, 2 (2019), 1930–1941.
[16] Omur Ozel, Kaya Tutuncuoglu, Jing Yang, Sennur Ulukus, and Aylin Yener. 2011. Transmission with energy harvesting nodes in fading wireless channels: Optimal policies. *IEEE J. Select. Areas Commun.* 29, 8 (2011), 1732–1743.
[17] Konstantinos Poularakis, Jaime Llorca, Antonia M. Tulino, Ian Taylor, and Leandros Tassiulas. 2020. Service placement and request routing in MEC networks with storage, computation, and communication constraints. *IEEE/ACM Trans. Netw.* 28, 3 (2020), 1047–1060.
[18] Samuel O. Somuyiwa, András György, and Deniz Gündüz. 2018. A reinforcement-learning approach to proactive caching in wireless networks. *IEEE J. Select. Areas Commun.* 36, 6 (2018), 1331–1344.
[19] Chuan Sun, Xiuhua Li, Junhao Wen, Xiaofei Wang, Zhu Han, and Victor C. M. Leung. 2023. Federated deep reinforcement learning for recommendation-enabled edge caching in mobile edge-cloud computing networks. *IEEE J. Select. Areas Commun.* 41, 3 (2023), 690–705.
[20] Sennur Ulukus, Aylin Yener, Elza Erkip, Osvaldo Simeone, Michele Zorzi, Pulkit Grover, and Kaibin Huang. 2015. Energy harvesting wireless communications: A review of recent advances. *IEEE J. Select. Areas Commun.* 33, 3 (2015), 360–381.
[21] Xiaoyu Xia, Feifei Chen, Qiang He, John Grundy, Mohamed Abdelrazek, and Hai Jin. 2020. Online collaborative data caching in edge computing. *IEEE Trans. Parallel Distrib. Syst.* 32, 2 (2020), 281–294.
[22] Han Xiao, Changqiao Xu, Yunxiao Ma, Shujie Yang, Lujie Zhong, and Gabriel-Miro Muntean. 2021. Edge computing-assisted multimedia service energy optimization based on deep reinforcement learning. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM'21)*. 1–6.
[23] Jie Xu, Lixing Chen, and Shaolei Ren. 2017. Online learning for offloading and autoscaling in energy harvesting mobile edge computing. *IEEE Trans. Cogn. Commun. Netw.* 3, 3 (2017), 361–373.
[24] Jie Xu, Lixing Chen, and Pan Zhou. 2018. Joint service caching and task offloading for mobile edge computing in dense networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'18)*. IEEE, 207–215.
[25] Zichuan Xu, Lizhen Zhou, Sid Chi-Kin Chau, Weifa Liang, Qiufen Xia, and Pan Zhou. 2020. Collaborate or separate? Distributed service caching in mobile edge clouds. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'20)*. IEEE, 2066–2075.
[26] Shizhe Zang, Wei Bao, Phee Lep Yeoh, Branka Vucetic, and Yonghui Li. 2019. Filling two needs with one deed: Combo pricing plans for computing-intensive multimedia applications. *IEEE J. Select. Areas Commun.* 37, 7 (2019), 1518–1533.
[27] Guanglin Zhang, Wenqian Zhang, Yu Cao, Demin Li, and Lin Wang. 2018. Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices. *IEEE Trans. Industr. Inf.* 14, 10 (2018), 4642–4655.

[28] Jing Zhang, Jun Du, Yuan Shen, and Jian Wang. 2020. Dynamic computation offloading with energy harvesting devices: A hybrid-decision-based deep reinforcement learning approach. *IEEE IoT J.* 7, 10 (2020), 9303–9317.
[29] Fengjun Zhao, Ying Chen, Yongchao Zhang, Zhiyong Liu, and Xin Chen. 2021. Dynamic offloading and resource scheduling for mobile-edge computing with energy harvesting devices. *IEEE Trans. Netw. Serv. Manage.* 18, 2 (2021), 2154–2165.

Received 15 October 2023; revised 26 December 2023; accepted 25 January 2024