User Preferences-Based Proactive Content Caching with Characteristics Differentiation in HetNets

Na Lin, Yamei Wang, Enchao Zhang, *Graduate Student Member, IEEE*, Shaohua Wan, *Senior Member, IEEE*, Ahmed Al-Dubai, *Senior Member, IEEE*, Liang Zhao, *Member, IEEE*,

***Abstract*—With the proliferation of mobile applications, the explosion of mobile data traffic imposes a significant burden on backhaul links with limited capacity in heterogeneous cellular networks (HetNets). To alleviate this challenge, content caching based on popularity at Small Base Stations (SBSs) has emerged as a promising solution. However, accurately predicting the file popularity profile for SBSs remains a key challenge due to variations in content characteristics and user preferences. Moreover, factors such as content size and the length of time slots (that is, the time duration of the update cycle for SBSs) critically impact the performance of caching schemes with limited storage capacity. In this paper, a realism-oriented intelligent caching (RETINA) is proposed to address the problem of content caching with unknown file popularity profiles, considering varying content sizes and time slots lengths. Our simulation results demonstrate that RETINA can significantly enhance the cache hit rate by 4%–12% compared to existing content caching schemes.**

***Index Terms*—Heterogeneous Cellular Networks, Reinforcement Learning, Content Caching, unknown file popularity profiles, content characteristics, user preferences.**

I. Introduction

With the evolution of communication technologies, the volume of mobile data traffic has witnessed a significant surge, particularly in video services [1]. Cisco projects that global mobile data traffic will expand nearly eight-fold by 2023 [2], placing considerable strain on backhaul links with limited capacity. Multimedia content, including movies, music and video clips, constitutes the predominant share, accounting for 80% of the total data traffic. To address the increasing demand for mobile data, the concept of Heterogeneous Cellular Networks (HetNets) has been introduced [3]. HetNets leverage a mix of Small Base Stations (SBSs) and Macro Base Stations (MBSs) to serve users over short-distance communication links, thereby enhancing network capacity and spectrum efficiency [4]–[6]. However, the cost associated with high-speed links connecting HetNets to the core network is exorbitant, and the user's Quality of Experience (QoE) is significantly impacted by connection quality. Furthermore, a notable trend is that a small subset of content generates the majority of mobile traffic. Consequently, caching popular content at SBSs has emerged as a viable strategy to enhance user QoE and alleviate backhaul pressure.

Efficient content caching at Small Base Stations (SBSs) necessitates the deployment of suitable caching strategies, which involve determining the content file to cache within the SBSs' limited caching capacity. The file popularity profile, representing the anticipated number of requests for each content file and indicative of user preferences, serves as a pivotal metric for designing optimal caching strategies [7]. However, in practical scenarios, user preferences are often unknown beforehand. Hence, the accuracy of predicting the file popularity profile emerges as a critical determinant of the caching scheme efficiency. In recent years, predicting file popularity profiles has garnered significant attention from both academia and industry. Due to the dynamic nature of file popularity, Reinforcement Learning (RL) has been proposed to learn file popularity online [8]–[13]. RL-based algorithms allow users to learn optimal caching strategies through interactions with the environment, facilitating real-time model updates. Consequently, RL presents a promising solution for content caching with time-varying file popularity. However, existing RL-based content caching schemes face challenges related to caching scheme performance and training efficiency, necessitating further research and development efforts.

First, existing research often overlooks the influence of content characteristics on the content caching process. Variations in content characteristics result in differences in the actual size of the content. For instance, multimedia files encompass diverse types such as movies, short videos, and music, each varying in size. Given the limited caching capacity of SBSs, the size of the content becomes a crucial factor in content caching decisions. When faced with files of similar popularity but varying sizes, prioritizing smaller files for caching conserves storage space, allowing SBSs to cache a greater number of popular files and maximize the overall rewards of the caching scheme. Thus, addressing the variance in content size across different file types emerges as a critical consideration.

Second, each SBS covers a distinct area in HetNet. Users located in different areas may prefer different types of files. Consequently, the consumption duration of each file type varies, significantly impacting the training efficiency of RL-based caching schemes. However, current RL-based caching schemes assume uniform time slot lengths across all SBSs, overlooking this variability. This assumption can lead to the following situations. On the one hand, the length of the time

Na Lin, Yamei Wang and Liang Zhao are with the School of Computer Science, Shenyang Aerospace University, Shenyang 110136, China. (e-mail: linna@sau.edu.cn; a13483024838@163.com; lzhao@sau.edu.cn).

Enchao Zhang is with the Graduate School of Informatics and Engineering, University of Electro-Communications, Tokyo, Japan. (e-mail: zhangenchaogm@gmail.com).

Shaohua Wan is with the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen 518110, China. (e-mail: shaohua.wan@ieee.org).

Ahmed Y.Al-Dubai is with the School of Computing, Edinburgh Napier University, UK. (e-mail: a.al-dubai@napier.ac.uk).

Liang Zhao is the corresponding author.

TABLE I
COMPARISON BETWEEN RELATED WORK AND OUR WORK

Studies	Heterogeneous network topology	Modeling caching problem	Content characteristics	The variety of user preferences in different regions
this work	multiple SBSs	CMAB	$\surd$	$\surd$
[18]-[20]	multiple SBSs	MDP	$\times$	$\times$
[21]	single SBS	CMAB	$\surd$	$\times$
[17],[22-30]	multiple SBSs	CMAB	$\times$	$\times$
[31],[33]	multiple SBSs	Contextual MAB	$\times$	$\times$
[32]	single SBS	Contextual MAB	$\times$	$\times$

slots is too short for a region. In the current time slot, most users still consume content from the prior slots, causing a large reduction in the number of available users (i.e. those who can request files during the request phase). Therefore, there is insufficient experience to learn knowledge of the file popularity profile, which decreases the training efficiency. On the other hand, the length of the time slots is too long. Most users have consumed the content requested in the current time slot and must wait for the arrival of the next time slot, increasing the total training costs. Hence, determining the time slot lengths for SBSs based on the consumption duration of file types preferred by users in the corresponding SBS coverage area is imperative for enhancing training efficiency and optimizing caching scheme performance.

To fill the aforementioned gaps, we propose a realism-oriented intelligent caching (RETINA) scheme for HetNet. RETINA leverages the RL-based algorithm to facilitate adaptive content caching without requiring prior knowledge, making it well-suited for predicting time-varying file popularity. Furthermore, RETINA comprehensively considers differences in content characteristics and user preferences during the content caching process. The key contributions of this paper are outlined as follows.

- To address the challenges posed by time-varying file popularity, we introduce an adaptive caching scheme called RETINA, designed to predict file popularity online. The proposed scheme efficiently delivers caching services with high effectiveness.
- Comprehensively addressing the impact of differences in content characteristics on caching system performance, RETINA integrates factors such as file popularity profiles and file sizes to generate caching strategies, thereby enhancing the overall performance of the caching scheme.
- To accommodate diverse user preferences across different regions, we propose a lightweight algorithm to adaptively determine the length of time slots for Small Base Stations (SBSs). This approach significantly enhances the training efficiency of RETINA.

The rest of the paper is structured as follows. We discuss related work in Section II. The system model and problem formulation are introduced in Section III. In Section IV, we present our realism-oriented intelligent caching scheme named RETINA and the analysis of RETINA. Section V shows simulation results. Section VI concludes the paper.

II. Related work

In this section, we summarize the existing work in the field of content caching with unknown file popularity profiles. We begin by summarizing the state-of-the-art trends in content caching. Next, we discuss the existing works by classifying them into two categories. Finally, we highlight the advantages of our proposed scheme, RETINA, by addressing the limitations of existing caching schemes.

A. Recent Trends in Content Caching

To address the increasing demand for mobile data, most recent works study the problem of caching popular content at SBSs to enhance user QoE and alleviate backhaul pressure. For instance, the works of [14]–[16] study the content placement problem in heterogeneous cellular caching networks with the aim of minimizing average user-experienced delay. The proposed schemes consider the stochastic arrival of user requests and traffic models, providing corresponding closed-form expressions. The author considers user mobility and proposes three novel mobile-aware caching schemes [15]. The work of [14] considers the overlapping coverage area of SBSs and proposes a Cooperative Most Popular Caching (CMPC) scheme. In [16], the author proposes a new analysis method based on queuing theory to find the minimum cache size for SBSs. However, these works overlook the dynamic nature of file popularity. In practice, file popularity is unknown and time-varying, critically impacting the performance of caching systems. Consequently, in recent years, most works have focused on the content caching problem of unknown file popularity and applied online RL to meet the time-varying file popularity profiles.

The RL-based algorithms allow users to learn optimal caching strategies online through interactions with the environment, facilitating real-time model updates. Accordingly, the caching strategy can be dynamically adjusted to keep up with trends of the time-varying file popularity profiles in real-time. Consequently, RL presents a promising solution for content caching with time-varying file popularity. The RL algorithms can be based on Markov decision processes (MDP) or Multi-armed bandits (MAB) [17]. MDP generally models a problem as a multi-state decision process, while MAB is usually a stateless (or single-state) model. The state-of-the-art works in MDP and MAB are discussed in the following subsections.

B. Markov Decision Processes-Based Schemes

MDP is normally a multi-state model (i.e., a problem is modeled as a multi-state decision process). The time-varying system state is a crucial factor in the decision-making process [18]. For instance, the authors of [19] consider actual time-varying channels and model them as a finite-state Markov channel. The caching problem is modeled to an MDP and the deep Q-learning algorithm is proposed to resolve the problem. The set of caching scheme states is a vector of content decision (i.e., whether the content is cached). In [20], the authors model the caching problem and joint pushing as an infinite-horizon average-cost MDP. The value function approximation of Q-learning and a deep Q-network are applied to resolve the problem. However, a common challenge in the MDP-based RL algorithms is the lack of a theoretical guarantee of performance, as the performance of the MDP-based algorithms cannot be mathematically proven to converge to an optimal value. Conversely, MAB offers a straightforward solution to content caching problems without the need for MDP modeling. Moreover, MAB-based algorithms are often more accessible for implementation in practice. Consequently, much of the research in wireless content caching has focused on MAB-based algorithms, as we review in the following.

C. Multi-armed bandit-Based Schemes

In the study of the prediction of file popularity profiles based on MAB, the prediction problem is modeled into two aspects, which are the stochastic combinatorial MAB (CMAB) problem and the stochastic contextual bandit problem. Most researchers study the prediction problem based on the stochastic CMAB problem [17], [21]–[30]. The work of [21] is the first work utilizing the MAB theory to resolve the problem of content caching with unknown file popularity profiles. The authors model the problem of content caching as a stochastic CMAB problem and design an online learning scheme to solve it. This work only sets content with different sizes but ignores the impact of the content size on the performance of the caching scheme during the process of content caching. In [29], their study particularly satisfies multiple small-cell scenarios. They consider that the network shares limited external resources, such as wireless backhaul resources and computational resources, and provide the intelligent scheme based on the stochastic CMAB with locked-up slots. In [26], the authors consider the scenario in which SBSs' coverage zones overlap. They model the problem of content caching as a stochastic CMAB problem with asleep arms and present an online learning scheme to address this problem, which users located in an overlapping area can freely select to connect to their preferred SBS. In [17], the authors consider users' availability and model the problem of content caching as a stochastic CMAB with asleep arms. Then, the authors provide an online learning scheme to address the problem. In their work, some arms are "asleep" (or unavailable) if they cannot provide feedback within the required time slot range.

With regard to the work of stochastic contextual bandits, in [31]–[33], the authors model the problem of content caching as a stochastic contextual MAB problem. The learning agent utilizes users' contextual information to learn the caching strategy, such as their gender, age, mood, and location. However, users do not want to divulge contextual information for their own privacy protection in real life. Therefore, the achievement of contextual information is unpractical to collect by network operators. Consequently, the problem transformation of the stochastic contextual bandit is unnecessary.

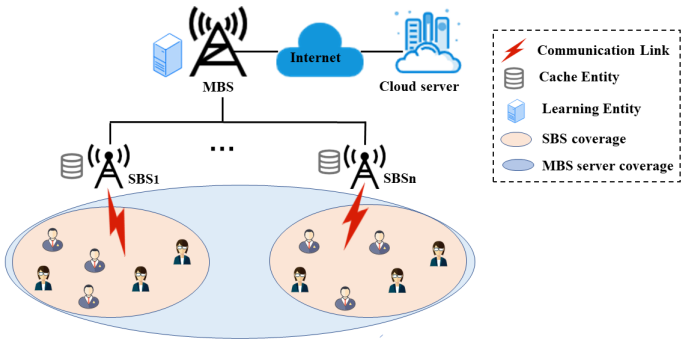


Fig. 1. Heterogeneous network topology.

D. Summary

Through extensive literature review in the field of content caching with unknown file popularity profiles, we discover the bottleneck issues in state-of-the-art works. First, most existing work ignores the impact of the differences in content characteristics during the process of content caching. Due to the differences in content characteristics, the actual size of the content is various. Considering the limited caching capacity of SBSs, the factor of content size plays an important role in content caching. Therefore, this paper considers the difference in the content's actual size when designing a caching scheme. Second, users located in different areas may prefer different types of files. Accordingly, the consumption duration of each type of file is different, which needs to be highlighted. However, in the existing RL-based caching work, the whole service time is slotted, and the length of each time slot is the same. This setup reduces the training efficiency and increases the training cost of RL. Consequently, it is necessary that the lengths of time slots for SBSs be determined according to the usage duration of file types preferred by users in the same SBS serving area. In our work, we leverage the MAB-based algorithm to learn file popularity profiles. We simultaneously consider both the difference in content characteristics and the variety of user preferences during the process of content caching. To the best of our knowledge, the proposed scheme is the first work to solve the problem of content caching with unknown file popularity profiles that jointly consider these two factors.

III. System Model and Problem Formulation

A. Network Model

We consider a heterogeneous cellular network where k cache-enabled SBSs are placed in the coverage area of an

MBS. As illustrated in Fig. 1, each SBS covers a disjointed area and works independently. Define the set of all SBSs as $\mathbb{K} := \{1, 2, ..., K\}$, and set $\mathbb{U}^k$ collects all users served by SBS k. Denote the set of all available files stored at the remote cloud server as $\mathbb{F}$ with size $|\mathbb{F}|$, by S_f the size of the f-th file in $\mathbb{F}$. All files are divided into n types, and the set of consumption duration of each file type is defined as $\mathbb{L} := \{l_1, l_2, ..., l_n\}$. Moreover, we utilize $\theta^k_{f,t}$ to express the popularity of file $f \in \mathbb{F}$ for SBS k at t-th time slot, and then the file popularity profile for SBS k is represented as $\Theta^k := \{\theta^k_f : \forall f \in \mathbb{F}\}$.

B. Cache Model

In the problem of content caching with unknown file popularity profiles, MBS acts as a learning agent to learn the file popularity profile and develops the caching strategy for SBSs. In addition, each SBS acts as a storage device with a capacity of $c < |\mathbb{F}|$ to store the favorite content of the served users. The MBS has higher computing power than the SBS, while the SBS is considerably closer to its users compared with the MBS. Obviously, this design can reduce backhaul traffic and energy consumption. Define p^k_t as the caching strategy of SBS k at t-th time slot. We define $\alpha^k_{f,t}$ as the caching state of file f for SBS k at t-th time slot.

$$\alpha^k_{f,t} = \begin{cases} 0, & \text{if the file f is not cached at SBS k} \\ 1, & \text{if the file f is cached at SBS k} \end{cases} \tag{1}$$

C. Request Model

For the user group $\mathbb{U}^k$, each of them sends the request independently and follows a Zipf distribution [34]. The popularity of the file depends on how many times it is requested. The probability of the j-th most popular content being requested is shown as,

$$p_j = \frac{1}{j^\gamma \sum_{i=1}^{N} i^{-\gamma}}, \quad where\ j \in \{1, 2, ..., \mathbb{F}\} \tag{2}$$

where p_j is the probability that the file *j* is requested by the users, *N* being the number of files, γ being the skewness of the content popularity profile. The larger the γ, the popularity profile becomes more skewed, i.e., the more concentrated on the high-ranked content requested by the users.

D. Service Model

The whole service time is divided into slots with slot index t = 1,2,..., T. In each slot, the following happens in sequence. (i) Each SBS reports the availability (see Remark) of users served to MBS, and MBS develops a caching strategy $p^k_t \subseteq \mathbb{F}$ for SBS k according to currently available users $u^k_t \subseteq \mathbb{U}^k$. (ii) MBS renews the SBS cache based on the caching strategy just determined. (iii) Each SBS broadcasts a list of cached content to its users, and all available users consume their favorite content. SBS then observes and reports the rewards of cached files to MBS. (iv) MBS updates the file popularity profile based on reports and adjusts the length of the next slot for SBSs.

TABLE II
NOTATIONS

Notation	Definition
$\mathbb{F}$	Collect all the Content
$\mathbb{K}$	Collect all the SBSs
S_f	The size of the f-th file in $\mathbb{F}$
$\mathbb{U}^k$	Collect the users served by SBS $k \in \mathbb{K}$
$\mathbb{L}$	The consumed time set of each file type
p^k_t	Caching strategy for SBS k in t-th time slot
u^k_t	Collect the available users for SBS k in t-th time slot
c	Caching size of the SBS
$\alpha^k_{f,t}$	The caching state of file f for SBS k at t-th time slot
γ	The skewness of the content popularity profile
$\theta^k_{f,t}$	The popularity of file $f \in \mathbb{F}$ for SBS k at t-th time slot
Θ^k	The file popularity profile for SBS k, i.e., $\Theta^k := \{\theta^k_f : \forall f \in \mathbb{F}\}$
$\mathbb{A}^k_t$	The number of each file requested by available users in the SBS k at t-th time slot
H^k_t	The cache hit rate of SBS k
Q	Request times queue
S	Request probability stock

Remark: The availability of users means that a user is available if he/she can request the cached files within the length of the current slot t, i.e., $[t, t+1]$.

E. Problem Formulation

In our work, each SBS has to cache the users' favorite files. Therefore, the target of the learning agent is to generate optimal caching strategies to maximize the cache hit rate of the caching scheme. Set $\mathbb{A}^k_t := \{A^k_{1,t}, A^k_{2,t}, ..., A^k_{f,t}\}$ is defined as the number of each file requested by available users in the SBS k at t-th time slot. And then the total number of requests from the available users of the SBS at t-th time slot is $|\mathbb{A}^k_t| := \sum_{i=1}^{f} A^k_{i,t}$. Hence, the cache hit rate of SBS k can be expressed as,

$$H^k_t = \frac{\sum_{i=1}^{f} A^k_{i,t} \cdot \alpha^k_{i,t}}{|\mathbb{A}^k_t|}. \tag{3}$$

The learning agent aims to maximize the cache hit rate of the caching scheme. Therefore, the problem is formulated as,

$$\max_{p^k_t}: \quad \sum_{t=1}^{T} \sum_{k=1}^{K} H^k_t$$

$$\text{s.t.} \quad p^k_t \subseteq F, \sum_{j \in p^k_t} S_j \leq c, \quad \forall k \in K. \tag{4}$$

For better readability, we summarize the notations as TABLE II.

IV. THE PROPOSED SCHEME

In this section, we present the details of RETINA to solve the above optimization problem. The proposed scheme tackles three primary challenges. The first challenge is uncertainty, as the file popularity profile Θ^k needs to be evaluated. Accordingly, we face a dilemma between exploration (estimating the reward of the other arm to reduce the influence of uncertainty) and exploitation (pulling the empirically best arms to achieve the highest instantaneous reward). In our scheme, the MAB-based algorithm is employed to balance exploitation and exploration, thereby maximizing the expected cumulative reward. The second challenge is differences in content characteristics, as the fact that the factor of content size critically impacts the performance of caching systems with limited storage capacity of SBSs. We improve the Upper Confidence Bound (UCB) algorithm to increase the performance of RETINA. The third challenge is the variety of user preferences in different regions, as the length of time slots impacts the performance of caching schemes. A lightweight algorithm is proposed to adaptively determine the length of time slots for SBSs, which significantly improves the training efficiency of RETINA. The specific implementation details are as follows in the subsections.

A. The balance of exploitation and exploration

In this subsection, the MAB-based algorithm is utilized to maximize the expected sum reward via exploration-exploitation management. We transform the content caching problem into the MAB, which refers to a player playing a slot machine with multiple arms. Every time the player pulls an arm, the machine gives him a reward from an unknown statistical function. The player aims to maximize the reward. In the beginning, the player knows nothing about the rewards of all the arms of the machine. It becomes a challenge to determine how to pull different arms within a limited number of attempts to obtain the greatest reward. To maximize the reward, the player explores to better exploit the arm. Therefore, we confront a dilemma between exploitation (pulling the empirically best arms to achieve the highest instantaneous reward) and exploration (estimating the reward of the other arm to reduce the influence of uncertainty). In MAB algorithms, the UCB algorithm [35] can better balance exploitation and exploration. Consequently, UCB is used to maximize the expected sum reward.

In the UCB learning model, agent, action, and reward need to be defined properly. In our caching scheme, MBS is regarded as the agent, and a file is treated as an arm. The caching decision of content f is treated as an action, which is defined as a_f (i.e., if content f is cached in the SBS $a_f = 1$ and if not $a_f = 0$). The reward is defined as the varying of the cache hit rate. For the SBS k, the reward in the t-th period is defined as,

$$R_t^k = H_t^k - H_{t-1}^k. \tag{5}$$

The UCB algorithm can achieve a relative balance of exploration and exploitation based on the currently known average reward of the rocker's arm plus a "boundary value". The core idea of UCB is to remain optimistic in the face of uncertainty by constructing a confidence radius around the estimation. We define the upper confidence bound $\overline{\theta}_{f,t}^k$ of the estimation for each arm (i.e., each file) as follows [35],

$$\overline{\theta}_{f,t}^k := \hat{\theta}_{f,N_{f,t}^k}^k + \sqrt{\frac{1.5 \ln t}{N_{f,t}^k}}, \tag{6}$$

where $N_{f,t}^k$ records the times that the file f has been chosen at the end of t-th time slot and $\hat{\theta}_{f,N_{f,t}^k}^k$ is the empirical mean reward (the average observed value) of the file f after all these $N_{f,t}^k$ observations. We call $\sqrt{\frac{1.5 \ln t}{N_{f,t}^k}}$ the confidence radius, and it is essentially the standard deviation of the mean. According to the reflection of the above formula, with the increasing value of the mean, the standard deviation is decreased, which further leads to the increasing chosen probability. Meanwhile, those arms that have been chosen fewer times will also obtain experimental opportunities. Mathematically, the true mean of $\hat{\theta}_f^k$ lies inside the confidence radius surrounding the empirical mean with a high likelihood according to Hoeffding's inequality [36]. If the file f has been chosen numerous times, it is intuitive that $\overline{\theta}_{f,t}^k$ is extremely close to the true mean θ_f^k.

When running the UCB algorithm, the values of $\hat{\theta}_{f,N_{f,t}^k}^k$ and $N_{f,t}^k$ need to be maintained for all $f \in \mathbb{F}$ respectively. At the end of t-th slot, the values of $\hat{\theta}_{f,N_{f,t}^k}^k$ and $N_{f,t}^k$ can be updated. Then, the rules to update these two values can be grouped into (7),

$$\begin{aligned} N_{f,t}^k &= N_{f,t-1}^k + 1 \\ \hat{\theta}_{f,N_{f,t}^k}^k &= \frac{N_{f,t-1}^k \hat{\theta}_{f,N_{f,t-1}^k}^k + \hat{\theta}_{f,N_{f,t}^k}^k}{N_{f,t}^k}. \end{aligned} \tag{7}$$

To comprehensively consider the impact of the difference in file size on the performance of the caching scheme, we add the file size to the indicator calculation formula $\overline{\theta}_{f,t}^k$. Specifically, we multiply $\overline{\theta}_{f,t}^k$ by a weight $\frac{1}{S_j}$(reciprocal of file j size), $j \in p_t^k$, marked as $\widetilde{\theta}_{f,t}^k$. It means the value per unit size. The advantage of this modification is that the MBS considers the file popularity profile and the actual size of files simultaneously when generating caching strategies for SBSs (i.e., selecting files with the maximum value of $\widetilde{\theta}_{f,t}^k$ instead of $\overline{\theta}_{f,t}^k$ at each slot), which can improve the performance of the caching scheme. $\widetilde{\theta}_{f,t}^k$ is defined in (8),

$$\widetilde{\theta}_{f,t}^k := \left(\hat{\theta}_{f,N_{f,t}^k}^k + \sqrt{\frac{1.5 \ln t}{N_{f,t}^k}} \right) \frac{1}{S_j}. \tag{8}$$

B. The adaptive setting of the length of time slots

1) The description of algorithm 1: Due to the variety of user preferences in different areas and the difference in file sizes, the lengths of time slots for SBSs be determined according to the usage duration of file types preferred by users in the same SBS serving area. The specific method is shown in Algorithm 1. We use LT to represent the length of the next

slot. Specifically, we set up a stack S^k for SBS k. The stack S^k stores n elements as $\{r_1, r_2, ..., r_n\}$, and r_n represents the probability that the n-th file type is requested. Meanwhile, we use queue Q to store the times of requests for each file type as $Q = \{q_1, q_2, ..., q_n\}$. Queue Q can be obtained by MBS. At the end of each slot, MBS calculates the value of r_n according to the user's request queue Q and sorts them in descending order, and then pushes them to S^k in turn, as shown in Lines 1 to 5. We use tp to represent the file type represented by the top element of the stack, i.e., $tp \in [1, n]$. Consequently, r_{tp} represents the value of the top element in the stack S. If r_{tp} is less than the specific value v, then it will be popped from the stack S, as shown in Lines 6 to 8. Finally, we judge whether stack S is empty. If stack S is not empty, then there must be at least one file type being requested that exceeds v. Then we set $LT = l_{tp}$. Conversely, the request probability of each file type is less than v, and then we set LT to one of the maximum values in the set $\mathbb{L}$, as shown in lines 9 to 13.

Algorithm 1 Length Of Next Slot

Input: Stack S: Store the probability of each file type being requested;
Request Queue Q: Store the times of requests for each type of file;
n: Number of file types;
v: The baseline to determine the length of the next time slot;
Output: LT: Length of the next slot;

1: Initial request queue $Q = \{q_1, q_2, ..., q_n\}$;
2: **for** $i = 1$ to n **do**
3: Calculate the probability of the i-th file type being requested based on $r_i = \frac{q_i}{\sum_{j=1}^{n} q_j}, \forall i \in [1, n]$;
4: **end for**
5: Sort $\{r_1, r_2, ..., r_n\}$ in descending order and Stack in turn;
6: **while** $r_{tp} < v$ **do**
7: pop;
8: **end while**
9: **if** Stock S != NULL **then**
10: $LT = l_{tp}$;
11: **else**
12: $LT = max(\mathbb{L})$;
13: **end if**

2) The determination of the value of v: in this subsection, we adaptively adjust the system setting to meet the time-varying user preference and further improve the training efficiency of RETINA. The value of v represents the baseline to determine the length of the next time slot. For instance, when v is set to 70%, if the request rate for a file type reaches 70%, the length of the next time slot is designed to the usage duration of this file type. The high value of v denotes the high request probability of a certain type. A large proportion of users' requirements are satisfied when the length of the next time slot is adjusted to the usage duration of this file type, which means that the number of available users increases at the next time slot. Therefore, RETINA has sufficient experience to learn knowledge of the file popularity profile, which improves the training efficiency of RETINA.

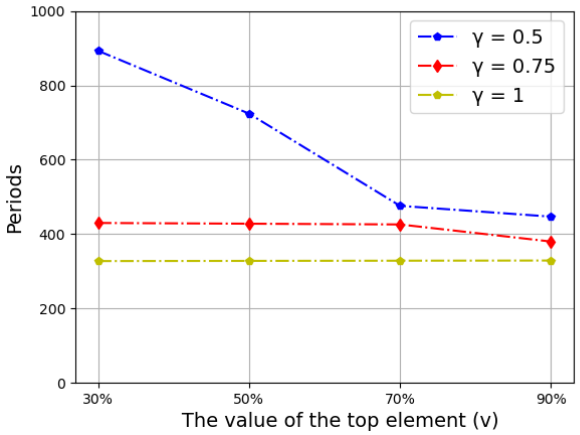


Fig. 2. The training efficiency of RETINA under different circumstances.

This condition denotes that the value of v has a critical impact on training efficiency. In addition, since the setting of the length of time slots is related to user preference, the value of v that can achieve optimal training efficiency is different under different user preference distributions. We can dynamically adjust this parameter according to different distributions of user preferences.

For different user preference distributions, we provide the simulation results on the training efficiency of RETINA under different values of v, which assist the determination of the value of v (as shown in Fig. 2). γ represents the degree of concentration of user request files, which reflects the largest user preference distribution. Three distributions of user preferences are set as $\gamma = 0.5, 0.75, 1$ to represent "unconcentrated", "relative concentration", and "quite concentration". As shown in Fig. 2, with the increase of γ, RETINA converges faster. The reason for this phenomenon is that if most users have the same preference, the variance of the data set and action space of RL is low, and the convergence speed to generate the optimal caching strategy is relatively fast.

When $\gamma = 0.5$, with the increased value of v, the training efficiency of RETINA is greatly improved. The reasons are as follows. First, when v is set to $[30\% - 50\%]$, due to the non-centralized distribution of user preferences, there may be a request rate of between 30% and 50% for a certain file type. The length of the next time slot is set to the consumption duration of this file type, which cannot meet the file usage duration of about 70% users. Consequently, the number of available users decreases at the next time slot, and RETINA has insufficient experience to learn knowledge of the file popularity profile, which reduces the training efficiency of RETINA. Second, when the value of v is set to a larger value, there is no file type with a request rate for the value of v. The length of the next time slot is set to the largest consumption duration of all file types, which can meet the file usage duration of most users. Consequently, the training efficiency of RETINA is improved. When $\gamma = 0.75$, due to the relative concentration of user preference distribution, there may be a request rate of 70% for a certain file type. The length

of the next time slot is set to the consumption duration of this file type, which cannot meet the file usage duration of about 30% users. Consequently, the training efficiency of RETINA is reduced. Under this preference distribution, there is no file type with a request rate of 90%. Consequently, the length of the next time slot is set to the largest consumption duration of all file types, which improves the training efficiency of RETINA. When $\gamma = 1$, the training efficiency of RETINA under different v values has a low variance. This is because the distribution of user preference is quite concentrated, and there may be a request rate of 90% for a certain file type. The length of the next time slot is set to the consumption duration of this file type, which can meet the file usage duration of most users. In conclusion, v can be set to about 90% when $\gamma = 0.5$ and $\gamma = 0.75$, and v can be set to any value between 10% and 100% when $\gamma = 1$.

Algorithm 2 The RETINA Scheme

1: **for** All $k \in K$ **do**
2: Initialize: cache all files at least once, observe the rewards, update $\hat{\theta}^k_{f,0}$ and $N^k_{f,0}$, and set $LT = \max(\mathbb{L})$;
3: **for** All $t = 1, 2, ..., T$ **do**
4: $N^k_{f,t} = N^k_{f,t-1}, \quad \forall f \in \mathbb{F}$;
5: $\hat{\theta}^k_{f,t} = \hat{\theta}^k_{f,t-1}, \quad \forall f \in \mathbb{F}$;
6: SBS observe the available users $u^k_t \subseteq \mathbb{U}^k$ and report to MBS;
7: MBS calculate the UCB estimate $\widetilde{\theta}^k_{f,t}, \forall f \in \mathbb{F}$ based on (8), and develop caching strategy p^k_t by resolving (4);
8: SBS play p^k_t;
9: MBS update $N^k_{f,t}$ and $\hat{\theta}^k_{f,N^k_{f,t}}$ base on (7);
10: Get LT by running **Algorithm 1** and adjust the length of the next slot to LT;
11: **end for**
12: **end for**

C. The overall caching scheme (RETINA)

Based on the above, the overall caching scheme called RETINA is presented in Algorithm 2. For SBS k, the content in the file library is cached at least once during initialization. Then SBS k observes the condition of the cached content to obtain the initially estimated content demand, and the MBS updates the parameters $\hat{\theta}^k_{f,N^k_{f,0}}$ and $N^k_{f,0}$, as shown in Lines 1 to 2. In t-th time slot, we use the values of $\hat{\theta}^k_{f,t}$ and $N^k_{f,t}$ gained at the end of time slot $t-1$ as the input of the current slot, as shown in Lines 4 to 5. The MBS observes the set of available users $u^k_t \subseteq \mathbb{U}^k$ and calculates the UCB estimate defined in (8) for each available user, and develops caching strategy p^k_t for SBS k. After that, the SBS refreshes the cache based on the caching strategy determined by MBS and broadcasts the cached content list to available users. All available users consume their favorite content, and then the SBS observes and reports the rewards of cached content to the MBS. Finally, the MBS updates $\hat{\theta}^k_{f,N^k_{f,t}}$ and $N^k_{f,t}$ for all arms in p^k_t according to actual observations and gets the length of the next slot LT by running algorithm 2, as shown in Lines 6 to 9. In this way, a stable caching strategy is obtained by continuous online learning.

D. The complexity analysis of RETINA

In this subsection, we analyze the time and space complexity, and the communication overhead for the proposed scheme. For the purpose of simplicity, we only analyze one SBS since each SBS works independently.

1) The time complexity analysis: In terms of time complexity, it appears that solving the optimization problem (4) consumes resources. first, an effective method to compute $\sum_{i \in p^k_t} \widetilde{\theta}^k_{f,t}$ for all j $\in \mathbb{F}$. Second, these values are sorted in descending order, and select the top c values as the optimal caching strategy. The time complexity of these values being sorted is $O(|\mathbb{F}|log|\mathbb{F}|)$.

2) The space complexity analysis: Regarding the space complexity, the MBS requires maintaining two values $N^k_{f,t}$ and $\hat{\theta}^k_{f,t}$ for each file. The space complexity is $O(|\mathbb{U}^k||\mathbb{F}|)$ for SBS k because there are $|\mathbb{U}^k||\mathbb{F}|$ pairs for SBS k. In addition, MBS needs to maintain the queue Q and stock S. In each slot, stock S and queue Q can buffer $\mathbb{T}$ values. In conclusion, the cost of maintaining the buffer is $O(|\mathbb{U}^k||\mathbb{F}|)$.

It indicates that our proposed RETINA scheme is efficient for the $O(|\mathbb{U}||\mathbb{F}|)$ space complexity and the $O(|\mathbb{F}|log|\mathbb{F}|)$ time complexity.

3) Communication overhead analysis: The length of each message used in communications is analyzed. There are four types of communication between the MBS, SBS, and users. Firstly, when the users report their availability to MBS by their connected SBS (i.e., a binary variable), the information cost for the communication is $O(|\mathbb{U}|)$. Secondly, MBS tells SBS to cache c content, and SBS broadcasts cached content to the users. Consequently, the information cost for the communication is $O(c)$. Thirdly, when the users request cached content, the information cost for the communication is $O(\mathbb{U})$. Finally, when SBS informs corresponding observational results to MBS, the information cost for the communication is $O(c)$.

As a result, the overall communication overhead of RETINA is $O(\mathbb{U} + c)$.

TABLE III
SIMULATION PARAMETERS AND VALUES

Parameter	Value
The number of files	1000
Size of each file	10-500 MB
Total size of the files	260000 MB
The caching capacity of SBS	16000 MB
The number of users	30, 50, 70
The number of file types	8
The resolution of the user's device	720P
The video code rate	$3Mb/s$

V. RESULTS AND DISCUSSION

A. Experiment Settings and Scenarios

In this section, RETINA is implemented to verify its performance. Experiments are conducted in the general scenario of

HetNet, where three SBSs are deployed within the coverage of one MBS. According to the simulation setting in [17], there are 30, 50, and 70 users respectively located in these three SBSs. We evaluate the performance of RETINA on video files. Note that we assume that the network environment and the data transmission rate of users are the same. We set the file amount to 1000 and the total file size to 260000 MB, i.e., $|\mathbb{F}| = 260000\ MB$. The caching capacity of each SBS is 16000 MB, i.e., $6\%|\mathbb{F}|$. The size of each file is set to vary randomly from 10 MB to 500 MB, which is derived from a realistic data set. We set the type of file amount to 8, and the required consumption duration of each type is given by the Eq. 9,

$$l_i = \frac{S_f \cdot 8}{r} \tag{9}$$

where r is the video code rate. We assume that the resolution of the user's device is $720P$, and r is set to $3Mb/s$. The Zipf distribution is used to model the request of users. Furthermore, Different values are set for experimental parameters, including the parameter of Zipf γ and the caching capacity of SBSs. The set of Zipf parameter γ is denoted as $\Gamma := [0, 0.5, 0.75, 1]$, and the set of caching capacities is denoted as $C := [5\%, 10\%, 15\%]$. We show the simulation parameters and the values in TABLE III.

B. Counterparts

To verify the performance of RETINA, we compare RETINA with the following schemes.

Optimal: Optimal has prior knowledge of file popularity profiles, which means that the popularity of the files requested by the user is known in advance. Under this assumption, Optimal can optimally cache the files requested by the user at the current moment and maximize the utility of the limited caching capacity, which is unrealistic. Optimal is only used to describe the upper bound of the performance of online learning schemes.

CFAUD: CFAUD is proposed in [17], the authors consider users' availability and delayed feedback. The authors model the problem of content caching as a stochastic CMAB with asleep arms and provide a UCB-based scheme to address the problem. In their work, some arms are "asleep" (or unavailable) if they cannot provide feedback within the required time slot range.

UCB-CS: UCB-CS is proposed in [26], the authors model the problem of content caching with unknown file popularity profiles under multiple SBSs as a stochastic CMAB problem and make use of the UCB algorithm to solve it. Our scheme fills some gaps in the UCB-CS scheme by considering the impact of the actual size of files on the caching scheme and the lengths of time slots.

$\epsilon-$greedy: In the $\epsilon-$greedy scheme, MBS randomly caches c files according to probability ϵ and caches c files with the highest empirical mean with probability $1-\epsilon$. In our experiment, we set $\epsilon = 0.1$.

SCCP: In [37], the authors devise a smart content caching policy (SCCP) for content caching in the heterogeneous cellular network. The authors adopt the cubic exponential smoothing method to predict the number of downloads of content downloaded by users.

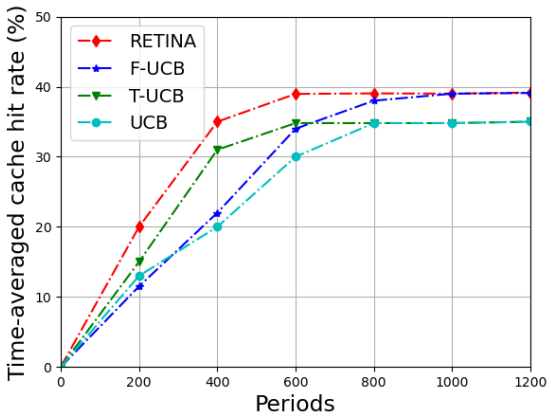


Fig. 3. Performance evaluation of RETINA and other UCB-based schemes.

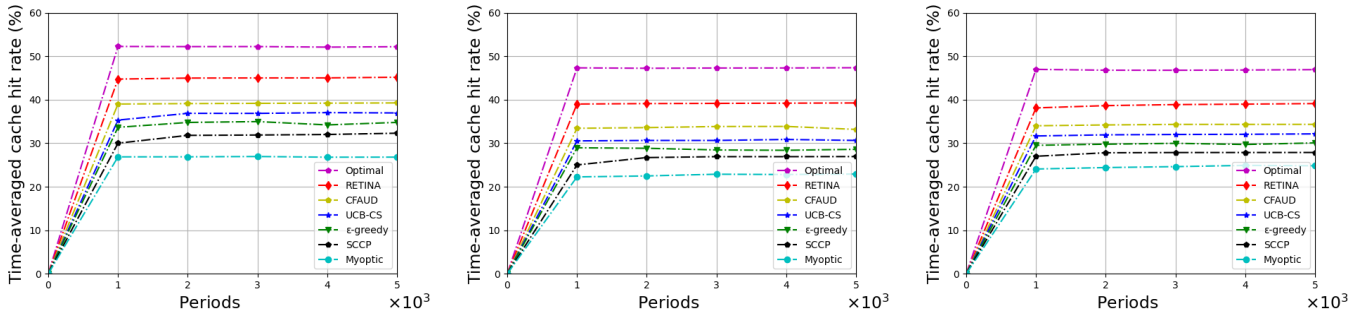
Myoptic: Myopic is that the agent will cache the popular content that was requested by users in the prior time slot. At the same time, the agent will randomly replace the content that has not been requested by users in the last time slot.

F-UCB: The UCB scheme that only considers the file size but not the length of time slots is called F-UCB.

T-UCB: The UCB scheme that only considers the length of time slots but not the file size is called T-UCB.

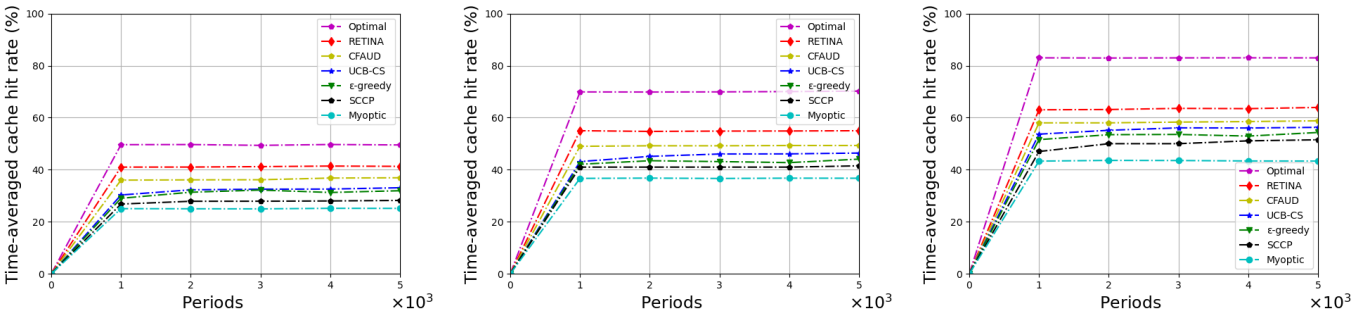
C. Performance Evaluation of RETINA

To assess the effect of the length of the files and time slots on the caching scheme, we have conducted extensive evaluations by vertical comparisons, as shown in Fig. 3. Fig. 3 presents the time-averaged cache hit rate of RETINA, F-UCB, T-UCB, and UCB. The comparison between RETINA and T-UCB reflects the impact of the file size on the caching scheme. The comparison between RETINA and F-UCB reflects the impact of the length of time slots on the caching scheme. First, RETINA and F-UCB consider the file size and add this indicator to our evaluation. The file popularity and the file size are considered simultaneously for the generation of caching strategies. However, T-UCB and UCB do not consider the file size. Consequently, the performance of RETINA and F-UCB has more advantages than T-UCB and UCB. Second, RETINA and T-UCB devise the adaptive setting of the length of time slots for SBSs, which can ensure sufficient knowledge for UCB to learn in each time slot. Consequently, RETINA and T-UCB can converge at a faster speed. However, F-UCB and UCB show disadvantages in terms of training efficiency since they cannot adjust the length of time slots adaptively. Third, RETINA and F-UCB have similar performance, but RETINA converges faster than F-UCB. In addition, RETINA and T-UCB have similar training efficiency, but the performance of RETINA is better than T-UCB. In conclusion, RETINA outperforms other UCB-based schemes in terms of cache hit rate and training efficiency because it comprehensively considers these two critical factors. The comparative evaluation



(a) SBS 1 (b) SBS 2 (c) SBS 3

Fig. 4. The time-averaged cache hit rate of Optimal, RETINA, CFAUD, UCB-CS, ϵ-greedy, SCCP, and Myoptic for different periods. We vary the number of users as $|\mathbb{U}^k| = 30, 50, 70$. Fig.4 (a) indicates experimental results as the $|\mathbb{U}^1| = 30$. Fig.4 (b) indicates experimental results as the $|\mathbb{U}^2| = 50$. Fig.4 (c) indicates experimental results as the $|\mathbb{U}^3| = 70$.



(a) $c = 5\%|\mathbb{F}|$ (b) $c = 10\%|\mathbb{F}|$ (c) $c = 15\%|\mathbb{F}|$

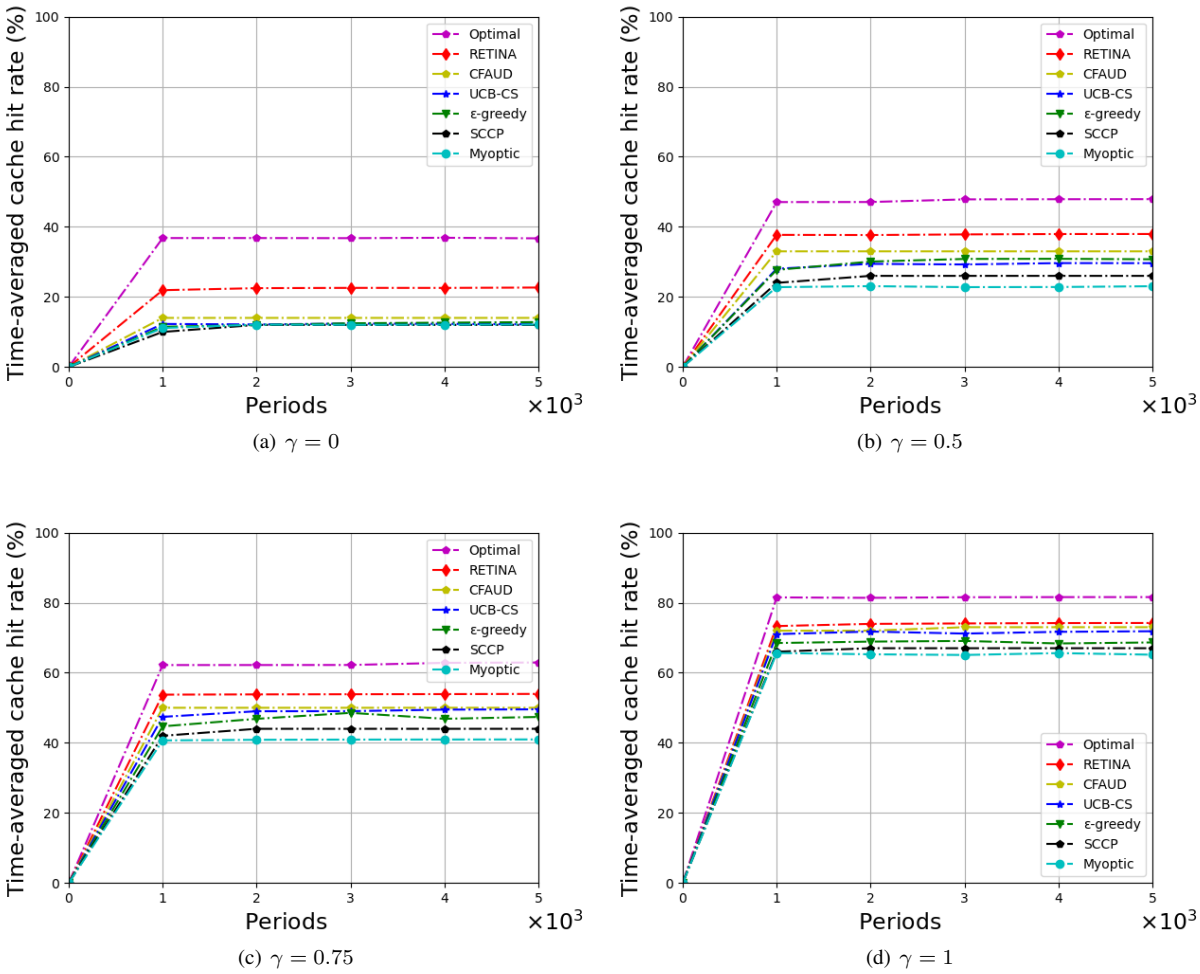
Fig. 5. The time-averaged cache hit rate of Optimal, RETINA, CFAUD, UCB-CS, ϵ-greedy, SCCP, and Myoptic for different periods. We vary the caching capacity of SBS as $c = 5\%|\mathbb{F}|, 10\%|\mathbb{F}|, 15\%|\mathbb{F}|$. Fig.5 (a) indicates experimental results as the $c = 5\%|\mathbb{F}|$. Fig.5 (b) indicates experimental results as the $c = 10\%|\mathbb{F}|$. Fig.5 (c) indicates experimental results as the $c = 15\%|\mathbb{F}|$.

demonstrates that RETINA can adaptively adjust its caching strategy with high effectiveness.

D. Comparative Performance Evaluation

We run all of the aforementioned schemes with three different numbers of user settings, i.e., $|\mathbb{U}^k| = 30, 50$, and 70. We run 5000 times for each simulation and take the average value as the simulation result. Simulation results are presented in Fig. 4. The time-averaged cache hit rate with 30 users, 50 users, and 70 users are shown in Fig. 4(a), Fig. 4(b), and Fig. 4(c) respectively. The Optimal has the maximum cache hit rate because it knows the expected number of requests for each file in advance and chooses the best content every time. The time-averaged cache hit rate of RETINA is better than that of CFAUD, UCB-CS, ϵ-greedy, SCCP, and Myoptic. The time-averaged cache hit rate of RETINA is 4% higher than that of the best-performing CFAUD among the reference schemes on average under different numbers of users. The reason for this phenomenon is that RETINA considers the impact of file size during the learning process, and the file size is added to the indicator calculation formula of UCB to maximize the time-averaged cache hit rate with the limited cache capacity of SBSs. In addition, the time-averaged cache hit rate of CFAUD and UCB-CS is larger than ϵ-greedy. This is because CFAUD and UCB-CS are closer to the exploitation stage, which effectively reduces the number of explorations. However, $\epsilon-$greedy only caches c files with probability ϵ and caches c files with the highest empirical mean with probability $1 - \epsilon$. The performance of UCB-based schemes is better than SCCP. The reason for this phenomenon is that SCCP is based on exponential smoothing. In exponential smoothing, the predicted value is the weighted sum of the previous observations. Consequently, the exponential smoothing has a lag effect, which indicates SCCP is insensitive to changes in user preference. In summary, due to the time-varying nature of file popularity, UCB-based caching schemes have more advantages than SCCP in dynamic scenarios. The learning ability of Myoptic is very weak since Myoptic only keeps the popular content that was requested by the user in the previous time slot. Consequently, Myoptic has the worst performance. As a consequence, RETINA is superior to CFAUD, UCB-CS, ϵ-greedy, SCCP, and Myoptic in different numbers of users.

We also analyze the effect of caching capacity on the time-averaged cache hit rate. The caching capacity of the SBS is set to $5\%|\mathbb{F}|, 10\%|\mathbb{F}|$, and $15\%|\mathbb{F}|$. Other parameters have not changed. Due to the independence and similarity of these three SBSs, we only demonstrate experiment results of SBS 1. Fig. 5 indicates the time-averaged cache hit rate of caching schemes



(a) $\gamma = 0$ (b) $\gamma = 0.5$

(c) $\gamma = 0.75$ (d) $\gamma = 1$

Fig. 6. The time-averaged cache hit rate of Optimal, RETINA, CFAUD, UCB-CS, ϵ-greedy, SCCP, and Myoptic for different periods. We change the parameter of Zipf as $\gamma = 0, 0.5, 0.75, 1$. Fig.6 (a) indicates experimental results when $\gamma = 0$. Fig.6 (b) indicates experimental results when $\gamma = 0.5$. Fig.6 (c) indicates experimental results when $\gamma = 0.75$. Fig.6 (d) indicates experimental results when $\gamma = 1$.

in various caching capacities. From Fig. 5(a), Fig. 5(b), and Fig. 5(c), we can observe that the time-averaged cache hit rate of each scheme increases as the caching capacity of the SBS increases. This is because SBS can cache more users' favorite files with the increase in cache capacity. Consequently, more users are able to access their favorite content via the SBS, thus increasing the hit rate of the caching scheme. There is no doubt that Optimal has the best performance. Besides, RETINA has better performance than CFAUD, UCB-CS, ϵ-greedy, SCCP, and Myoptic in various caching capacities. The time-averaged cache hit rate of RETINA is on average 4% higher than that of the best-performing CFAUD among the reference schemes under various caching capacities of SBS. The reason for this phenomenon is that RETINA adds the file size to the indicator calculation formula. Consequently, RETINA makes the time-averaged cache hit rate as large as possible at the limited caching capacity of SBS. To sum up, RETINA is suitable for various caching capacities of the SBS.

We further assess the impact of parameter γ on the time-averaged cache hit rate, and the test set is $\Gamma := [0, 0.5, 0.75, 1]$. γ is the skewness of the content popularity profile, which represents the degree of concentration of user request files. Consistent with the above, other parameters have not changed, and we only analyze SBS 1. Simulation results are indicated in Fig. 6. Figs. 6(a), 6(b), 6(c), and 6(d) show the time-averaged cache hit rate of Optimal, RETINA, CFAUD, UCB-CS, ϵ-greedy, SCCP, and Myoptic in different popularity parameters for different periods. As shown in Fig.6, with the increasing value of γ, the performance of these caching schemes is continuously improved. The reason for this phenomenon is that file requests are increasingly focused on a small number of files, which indicates that the estimation of file popularity is critical. When $\gamma = 0$, it indicates that the content requested by users is relatively scattered. Because of the limited caching capacity of SBSs, each online learning scheme does not perform very well. However, RETINA is better than CFAUD, UCB-CS, ϵ-greedy, SCCP, and Myoptic. The time-averaged cache hit rate of RETINA is 10% better than that of the best-performing CFAUD among the reference schemes as shown in Fig. 6(a). The reason for this phenomenon is that RETINA considers the impact of file size on performance. When the file requested by users is relatively scattered, the size of the files is important.

In addition, we can see that the performance of CFAUD and UCB-CS is better than ϵ-greedy and Myoptic. This is because CFAUD and UCB-CS are closer to the exploitation stage with the increase of γ and are more dependent on the estimation of content demand. When $\gamma = 1$, the performance of RETINA is comparable to that of CFAUD, which is the result in Fig. 6(d). This is because the caching capacity of SBSs may not be less than the size of the popular content as the requested content is more and more concentrated on a very small amount of content. To sum up, the applicable γ value range of RETINA is [0, 1), and this range is also of practical significance.

VI. CONCLUSION

In this paper, we conduct an extensive literature review in the field of content caching based on the prediction of file popularity profiles. The existing content caching schemes have various gaps and face challenges that should be addressed because of the variety of user preferences and the differences in content characteristics. To progress the state-of-the-art works, we comprehensively address these challenges and propose a realism-oriented intelligent caching scheme named RETINA, aimed at maximizing cache hit rates. In the evaluation section, we compare RETINA against other content caching schemes to assess its training efficiency and scalability. Simulation results indicate that RETINA can increase the cache hit rate by 4%–12% compared with its counterparts.

ACKNOWLEDGMENT

This paper is supported in part by the Liaoning Provincial Department of Education Science Foundation under Grant LJKZ0206(Key Projects), in part by the Natural Science Foundation of Liaoning Province under Grant 2021-BS-190, and in part by the Liaoning Provincial Department of Education Science Foundation under Grant JYT2020046.

REFERENCES

[1] H. Li, M. Sun, F. Xia, X. Xu, and M. Bilal, "A survey of edge caching: Key issues and challenges," *Tsinghua Science and Technology*, vol. 29, no. 3, pp. 818–842, 2024.

[2] U. Cisco, "Cisco annual internet report (2018–2023) white paper," *Online](accessed March 26, 2021) https://www. cisco. com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/whitepaper-c11-741490. html*, 2020.

[3] A. Ghosh, N. Mangalvedhe, R. Ratasuk, B. Mondal, M. Cudak, E. Visotsky, T. A. Thomas, J. G. Andrews, P. Xia, H. S. Jo *et al.*, "Heterogeneous cellular networks: From theory to practice," *IEEE communications magazine*, vol. 50, no. 6, pp. 54–64, 2012.

[4] Y. Zhang, M. A. Kishk, and M.-S. Alouini, "Computation offloading and service caching in heterogeneous mec wireless networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 6, pp. 3241–3256, 2023.

[5] T. Cao, Z. Zhang, X. Wang, H. Xiao, and C. Xu, "Ptcc: A privacy-preserving and trajectory clustering-based approach for cooperative caching optimization in vehicular networks," *IEEE Transactions on Sustainable Computing*, pp. 1–16, 2024.

[6] H. Li, X. Li, C. Sun, F. Fang, Q. Fan, X. Wang, and V. C. M. Leung, "Intelligent content caching and user association in mobile edge computing networks for smart cities," *IEEE Transactions on Network Science and Engineering*, vol. 11, no. 1, pp. 994–1007, 2024.

[7] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: design aspects, challenges, and future directions," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 22–28, 2016.

[8] D. Qiao, S. Guo, D. Liu, S. Long, P. Zhou, and Z. Li, "Adaptive federated deep reinforcement learning for proactive content caching in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4767–4782, 2022.

[9] Z. Wang, J. Hu, G. Min, Z. Zhao, and Z. Wang, "Agile cache replacement in edge computing via offline-online deep reinforcement learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 35, no. 4, pp. 663–674, 2024.

[10] X. Liu, M. Derakhshani, and S. Lambotharan, "Contextual learning for content caching with unknown time-varying popularity profiles via incremental clustering," *IEEE Transactions on Communications*, vol. 69, no. 5, pp. 3011–3024, 2021.

[11] Y. Qian, R. Wang, J. Wu, B. Tan, and H. Ren, "Reinforcement learning-based optimal computing and caching in mobile edge network," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2343–2355, 2020.

[12] Z. Yang, Y. Liu, Y. Chen, and L. Jiao, "Learning automata based q-learning for content placement in cooperative caching," *IEEE Transactions on Communications*, vol. 68, no. 6, pp. 3667–3680, 2020.

[13] J. Shi, L. Zhao, X. Wang, W. Zhao, A. Hawbani, and M. Huang, "A novel deep q-learning-based air-assisted vehicular caching scheme for safe autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4348–4358, 2020.

[14] F. Rezaei, B. H. Khalaj, M. Xiao, and M. Skoglund, "Performance analysis of heterogeneous cellular caching networks with overlapping small cells," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, pp. 1941–1951, 2022.

[15] S. A. A. Siahpoosh and F. Rezaei, "A study on the impact of mobility on caching in non-standalone 5g vehicular networks," *Vehicular Communications*, vol. 41, no. Jun., pp. 1–11, 2023.

[16] F. Rezaei, B. H. Khalaj, M. Xiao, and M. Skoglund, "Delay and stability analysis of caching in heterogeneous cellular networks," in *2016 23rd International Conference on Telecommunications (ICT)*, 2016, pp. 1–5.

[17] Z. Huang, B. Hu, and J. Pan, "Caching by user preference with delayed feedback for heterogeneous cellular networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1655–1667, 2020.

[18] F. Imani, Z. Qiu, and H. Yang, "Markov decision process modeling for multi-stage optimization of intervention and treatment strategies in breast cancer," in *2020 42nd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*, 2020, pp. 5394–5397.

[19] Y. He, Z. Zhang, F. R. Yu, N. Zhao, H. Yin, V. C. Leung, and Y. Zhang, "Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 11, pp. 10 433–10 445, 2017.

[20] Z. Lin, W. Huang, and W. Chen, "Bandwidth and storage efficient caching based on dynamic programming and reinforcement learning," *IEEE Wireless Communications Letters*, vol. 9, no. 2, pp. 206–209, 2020.

[21] P. Blasco and D. Gündüz, "Learning-based optimization of cache content in a small cell base station," in *2014 IEEE International Conference on Communications (ICC)*. IEEE, 2014, pp. 1897–1903.

[22] X. Xu and M. Tao, "Decentralized multi-agent multi-armed bandit learning with calibration for multi-cell caching," *IEEE Transactions on Communications*, vol. 69, no. 4, pp. 2457–2472, 2021.

[23] L. Su, R. Zhou, N. Wang, J. Chen, and Z. Li, "Multi-agent multi-armed bandit learning for content caching in edge networks," in *2022 IEEE International Conference on Web Services (ICWS)*, 2022, pp. 11–16.

[24] Y. Han, L. Ai, R. Wang, J. Wu, D. Liu, and H. Ren, "Cache placement optimization in mobile edge computing networks with unaware environment—an extended multi-armed bandit approach," *IEEE Transactions on Wireless Communications*, vol. 20, no. 12, pp. 8119–8133, 2021.

[25] X. Xu, M. Tao, and C. Shen, "Collaborative multi-agent multi-armed bandit learning for small-cell caching," *IEEE Transactions on Wireless Communications*, vol. 19, no. 4, pp. 2570–2585, 2020.

[26] B. Hu, Y. Chen, Z. Huang, N. A. Mehta, and J. Pan, "Intelligent caching algorithms in heterogeneous wireless networks with uncertainty," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 1549–1558.

[27] G. Xiong, S. Wang, G. Yan, and J. Li, "Reinforcement learning for dynamic dimensioning of cloud caches: A restless bandit approach," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 2108–2117.

[28] ——, "Reinforcement learning for dynamic dimensioning of cloud caches: A restless bandit approach," *IEEE/ACM Transactions on Networking*, pp. 1–15, 2023.

[29] B. Hu, M. Tanha, D. Sajjadi, and J. Pan, "Intelligent caching in dense small-cell networks with limited external resources," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*. IEEE, 2018, pp. 295–298.

[30] Y. Miao, Y. Hao, M. Chen, H. Gharavi, and K. Hwang, "Intelligent task caching in edge cloud via bandit learning," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 1, pp. 625–637, 2020.

[31] Y. Huang, P. Dai, K. Zhao, and H. Xing, "Contextual multi-armed bandit learning for freshness-aware cache update in vehicular edge networks," in *2022 IEEE International Symposium on Product Compliance Engineering - Asia (ISPCE-ASIA)*, 2022, pp. 1–6.

[32] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 2, pp. 1024–1036, 2016.

[33] C. Dai, K. Zhu, R. Wang, and B. Chen, "Contextual multi-armed bandit for cache-aware decoupled multiple association in udns: A deep learning approach," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 4, pp. 1046–1059, 2019.

[34] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*, vol. 1. IEEE, 1999, pp. 126–134.

[35] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2, pp. 235–256, 2002.

[36] W. Hoeffding, "Probability inequalities for sums of bounded random variables," in *The collected works of Wassily Hoeffding*. Springer, 1994, pp. 409–426.

[37] Y. Li, H. Ma, L. Wang, S. Mao, and G. Wang, "Optimized content caching and user association for edge computing in densely deployed heterogeneous networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 6, pp. 2130–2142, 2022.

Na Lin received the M.S. degree in computer science from the Shenyang University of Technology, China, in 2001, and the Ph.D. degree in computer science from Northeastern University, China, in 2005. From 2006 to 2010, she worked as a Postdoctoral Researcher in Northeastern University. She is currently a Professor with the School of Computer Science, Shenyang Aerospace University, China. She was a visiting scholar at the University of Leicester, UK, from 2019 to 2020. Her research interests include air-ground integrated network, mobile edge computing, intelligent transportation, FANET, UAV swarm and SDVN, etc.

Yamei Wang received the B.S. degree in computer science and technology from Hebei Normal University Of Science & Technology, China. She is currently working toward the master's degree in computer technology with the School of Computer Science, Shenyang Aerospace University, China. Her research interests mainly include MEC, edge caching, and reinforcement learning.

Enchao Zhang (Graduate Student Member, IEEE) is currently pursuing his Ph.D. degree with the Graduate School of Informatics and Engineering, University of Electro-Communications in Tokyo, Japan. In 2023, he obtained his M.S. degree from Shenyang Aerospace University in China, where his graduation thesis was recognized as the "Outstanding Master's Graduation Thesis in Liaoning Province". He is the recipient of the "Best Paper Award" at IEEE EUC-2022. His current research interests mainly include network communication, edge computing, quantum computing and quantum AI.

Shaohua Wan (Senior Member, IEEE) received the Ph.D. degree from the School of Computer, Wuhan University in 2010. Since 2010, he has been an associate professor with the School of Information and Safety Engineering, Zhongnan University of Economics and Law. From 2016 to 2017, he was a visiting professor at with the Department of Electrical and Computer Engineering, Technical University of Munich, Germany. His main research interests include deep learning for Internet of Things and edge computing. He is an author of over 100 peer-reviewed research papers and books, including over 40 IEEE/ACM Transactions papers such as TII, TITS, TOIT, TNSE, TMM, TCSS, TOMM, PR, etc., and many top conference papers in the fields of Edge Intelligence.

Ahmed Y. Al-Dubai is Professor of Networking and Communication Algorithms in the School of Computing at Edinburgh Napier University, UK. He received the PhD degree in Computing from the University of Glasgow in 2004. His research interests include Communication Algorithms, Mobile Communication, Internet of Things, and Future Internet. He received several international awards.

Liang Zhao (S'09–M'17) is a Professor at Shenyang Aerospace University, China. He received his Ph.D. degree from the School of Computing at Edinburgh Napier University in 2011. Before joining Shenyang Aerospace University, he worked as associate senior researcher in Hitachi (China) Research and Development Corporation from 2012 to 2014. He is also a JSPS invitational Fellow (2023). He was listed as Top 2% of scientists in the world by Standford University (2022). His research interests include ITS, VANET, WMN and SDN. He has published more than 150 articles. He served as the Chair of several international conferences and workshops, including 2022 IEEE BigDataSE (Steering Co-Chair), 2021 IEEE TrustCom (Program Co-Chair), 2019 IEEE IUCC (Program Co-Chair), and 2018-2022 NGDN workshop (founder). He is Associate Editor of Frontiers in Communications and Networking and Journal of Circuits Systems and Computers. He is/has been a guest editor of IEEE Transactions on Network Science and Engineering, Springer Journal of Computing, etc. He was the recipient of the Best/Outstanding Paper Awards at 2015 IEEE IUCC, 2020 IEEE ISPA, 2022 IEEE EUC, and 2013 ACM MoMM.