Cooperative Edge Caching Based on Elastic Federated and Multi-Agent Deep Reinforcement Learning in Next-Generation Networks

Qiong Wu, *Senior Member, IEEE*, Wenhua Wang, Pingyi Fan, *Senior Member, IEEE*, Qiang Fan, Huiling Zhu, *Senior Member, IEEE* and Khaled B. Letaief, *Fellow, IEEE*

***Abstract*—Edge caching is a promising solution for next-generation networks by empowering caching units in small-cell base stations (SBSs), which allows user equipments (UEs) to fetch users' requested contents that have been pre-cached in SBSs. It is crucial for SBSs to predict accurate popular contents through learning while protecting users' personal information. Traditional federated learning (FL) can protect users' privacy but the data discrepancies among UEs can lead to a degradation in model quality. Therefore, it is necessary to train personalized local models for each UE to predict popular contents accurately. In addition, the cached contents can be shared among adjacent SBSs in next-generation networks, thus caching predicted popular contents in different SBSs may affect the cost to fetch contents. Hence, it is critical to determine where the popular contents are cached cooperatively. To address these issues, we propose a cooperative edge caching scheme based on elastic federated and multi-agent deep reinforcement learning (CEFMR) to optimize the cost in the network. We first propose an elastic FL algorithm to train the personalized model for each UE, where adversarial autoencoder (AAE) model is adopted for training to improve the prediction accuracy, then a popular content prediction algorithm is proposed to predict the popular contents for each SBS based on the trained AAE model. Finally, we propose a multi-agent deep reinforcement learning (MADRL) based algorithm to decide where the predicted popular contents are collaboratively cached among SBSs. Our experimental results demonstrate the superiority of our proposed scheme to existing baseline caching schemes.**

***Index Terms*—cooperative edge caching, elastic federated learning, multi-agent deep reinforcement learning, next-generation networks**

I. Introduction

This work was supported in part by the National Natural Science Foundation of China under Grant 61701197, in part by the National Key Research and Development Program of China under Grant 2021YFA1000500(4), in part by the Research Grants Council under the Areas of Excellence scheme grant AoE/E-601/22-R, and in part by 111 Project under Grant B23008. (Corresponding author: Pingyi Fan.)

Qiong Wu and Wenhua Wang are with the School of Internet of Things Engineering, Jiangnan University, Wuxi 214122, China. (e-mail: qiongwu@jiangnan.edu.cn, wenhuawang@stu.jiangnan.edu.cn)

Pingyi Fan is with the Department of Electronic Engineering, Beijing National Research Center for Information Science and Technology, Tsinghua University, beijing 100084, China (Email: fpy@tsinghua.edu.cn)

Qiong Fan is with Qualcomm, San Jose, CA 95110, USA (e-mail: qf9898@gmail.com)

Huiling Zhu is with the School of Engineering, University of Kent, CT2 7NT Canterbury, U.K.(e-mail: H.Zhu@kent.ac.uk)

K. B. Letaief is with the School of Engineering, Hong Kong University of Science and Technology (HKUST), Hong Kong (email: eekhaled@ust.hk)

In recent years, with the increasing popularity of smart devices, we have witnessed an unprecedented growth in mobile data traffic, which has imposed a heavy burden on wireless networks [1]–[3]. As users increasingly rely on user equipment (UEs) like mobile devices and home routers to access content from wireless networks, it becomes challenging for ensuring a satisfactory quality of service to meet their demands [4]–[6]. To address this challenge, edge caching has emerged as a promising solution for next-generation networks [7]–[9]. Through the implementation of caching units in wireless edge nodes, such as small-cell base stations (SBSs), UEs can fetch users' desired contents from nearby SBSs instead of remote servers or cloud. This process significantly reduces traffic load, alleviates network congestion, reduces latency and improves system performance [10].

To efficiently enable UEs to fetch users' desired contents from SBSs, SBSs need to predict popular contents based on the interests of users within their coverage area and proactively cache these popular contents in advance. Due to the unique preference of each user, the popular contents cached in different SBSs may exhibit surprising variations [11]. Machine learning (ML) can overcome this issue by training based on users' data to extract hidden features, thereby effectively predicting popular contents [12]. However, most traditional ML algorithms need to train and analyze the data generated by users, which may involve personal sensitive information. Federated Learning (FL) has emerged as a potential solution that can protect the privacy and security of users' data [13], [14]. For the traditional FL, each UE trains its local model based on its users' data, and then only uploads the trained local model to the local SBS instead of its training data. Then the local SBS aggregates the uploaded models from UEs to update the global model. However, traditional FL tends to generalize user preferences, neglecting the personalized patterns of individual UEs. This general approach may not cater to the specific needs and behaviors of individual UEs, leading to potential inefficiencies in someway. To address this limitation, we introduce the elastic FL algorithm which adjusts the learning process dynamically based on the unique characteristics of each UE. By assigning specific weights to each local model based on differences between the global and local models, one can capture users' individual characteristics [15]. This elastic adjustment ensures that the final model is both globally informed and locally sensitive. This dual feature significantly enhances both prediction accuracy and caching

hit ratios in edge caching scenarios.

After each local SBS has predicted the popular contents, deploying edge caches poses an additional challenge. In next-generation networks, the importance of collaborative caching is further strengthened. Here, cached contents can be shared among adjacent SBSs via the Xn interface [16]. This allows UEs to indirectly obtain the requested contents from the adjacent SBSs even if they cannot directly fetch the requested contents from the local SBS. The majority of research in collaborative caching has focused on joint optimization with metrics like resource allocation and latency reduction, rather than concentrating on popular contents. Additionally, while studies employing artificial intelligence technology generally use popular contents in their caching strategies, they often overlook the crucial aspect of predicting content popularity for future proactive caching. Therefore, predicting future popular contents and utilizing them for collaborative caching can significantly enhance caching efficiency. However, the number of predicted popular contents for each SBS is usually larger than the limited cache capacity of the SBS. Therefore, this requires cooperation between SBSs to cache popular contents. Additionally, the placement of these popular contents in the right SBSs is also critical, as it directly impacts the cost of fetching contents across the network. In addition, the dimensions of caching decision will increase with the number of SBSs and cached contents, inevitably adding complexity to the caching problem in the entire network. This makes it challenging to coordinate caching among SBSs to reduce the cost for fetching contents in the next-generation network. Multi-agent reinforcement learning (MADRL) is a suitable method that can overcome the above problem, and has begun to be used in cooperative edge caching. Instead of the traditional way of making separate decisions, MADRL promotes cooperation among SBSs. Each SBS, which is considered an agent, does not just decide where to cache popular contents based on its own prediction. Instead, it should work together with other SBSs, taking into account their decisions to create a network-wide caching strategy driven by content popularity. If some UEs' demands change, MADRL can adjust the caching decisions. This adaptability is crucial for handling the increased complexity that comes with the growing caching needs in the next-generation network.

In this paper, we propose a cooperative edge caching scheme based on elastic federated learning and multi-agent deep reinforcement learning (CEFMR) scheme to optimize the cost of fetching contents in the next-generation network. The contributions of this paper are summarized as follows[1]

1) Each SBS employs an adversarial autoencoder (AAE) model to predict the content popularity within its own coverage. It helps to learn deep latent representations from users' historical requests and contextual information, enabling the discovery of implicit relationships between users and contents, thus improving prediction accuracy.
2) We propose an elastic FL algorithm to train the AAE model for each SBS, which assigns specific weight for each UE's model based on the differences between the global model and local model, thereby it can protect users' privacy and obtain the personalized local model for each UE.
3) We formulate the collaborative edge caching problem as a caching decision problem based on the MADRL framework, where states, actions, and reward have been defined. This formulation aims to optimize the cost for fetching contents in the network. Afterwards, we propose the multi-agent deep deterministic policy gradient (MADDPG) based algorithm to learn the optimal caching decision.

The remainder of this paper is structured as follows. Section II reviews the related works on the research on popularity prediction and collaborative caching. Section III provides a concise overview of the system model. Section IV introduces the proposed cooperative edge caching scheme. We present some simulation results in Section V, and then conclude them in Section VI.

II. Related Work

We first reviewed the research on popularity prediction and collaborative caching, and then we reviewed the studies on collaborative caching based on MADRL and FL.

Currently, there are studies on content popularity prediction. In [17], Pervej *et al.* proposed a novel approach to edge caching in next-generation wireless networks using a long short-term memory based sequential model to dynamically predict user preferences for content caching. Additionally, they formulated a non-convex optimization problem to minimize content sharing costs and employed a greedy algorithm to find a sub-optimal solution. In [18], Yao *et al.* introduced vehicular content centric network to address mobility challenges in vehicular ad hoc networks by implementing content centric networking. They proposed popularity-based content caching, a cache replacement scheme that predicts content popularity using hidden Markov Model based on received interests, request ratio, frequency and content priority. In [19], Tao *et al.* proposed a Gaussian process based Poisson regressor to accurately and efficiently model content request patterns. Bayesian learning is employed to robustly learn model parameters, and to handle the lack of closed-form expression for the posterior distribution, a Stochastic Variance Reduced Gradient Hamiltonian Monte Carlo method is applied. In [20], Liu *et al.* presented a content popularity prediction algorithm based on auto-regressive models for Information-Centric Networking in the Internet of Things, improving cache hit rates, reducing network traffic, and minimizing service access delays, particularly beneficial for real-time streaming media services. In [21], Feng *et al.* introduced a content popularity prediction algorithm, GRU-Attention, tailored for Vehicle Named Data Networking, which relied on in-network caching to swiftly serve subsequent content requests. By leveraging the attention mechanism and GRU model, it accurately forecasted future popular contents based on multiple historical request patterns,

[1]This paper has been submitted to IEEE TNSM. The source code has been released at: https://github.com/qiongwu86/Edge-Caching-Based-on-Multi-Agent-Deep-Reinforcement-Learning-and-Federated-Learning

enhancing prediction accuracy and optimizing cache utilization within the network.

There are already studies on collaborative caching. In [22], Pervej *et al.* tackled the optimization of collaborative caching in heterogeneous edge networks by developing an improved Particle Swarm Optimization algorithm, demonstrating through numerical analysis and simulations that this approach significantly enhances cache hit ratios compared to traditional caching strategies. In [23], Saputra *et al.* employed a cooperative mobile edge caching network by jointly optimizing collaborative caching and routing decisions to reduce content access delays and backhaul traffic. To address the optimization challenge, they transformed the problem into a mixed-integer nonlinear programming problem and utilized a branch-and-bound algorithm with the interior-point method to find a near-optimal solution. In [24], Xie *et al.* proposed a cooperative caching scheme where user terminals cache popular services to an intelligent routing relay by using an online reverse auction combined with a first-come-first-served strategy. In [25], Li *et al.* employed a crowdsourced approach for cooperative caching in Content-Centric Mobile Networking. They introduced a robust caching control scheme using Kullback-Leibler divergence to manage uncertainties in user mobility, finally formulated as a chance-constrained robust optimization problem. In [26], Liao *et al.* employed cooperative caching strategies using maximum distance separable codes for content restructuring and optimizes both content placement and cooperation policies among base stations. A compound caching strategy, namely multicast-aware cooperative caching assuming fixed and dynamic cooperative policies, respectively, was developed to combine the merits of multicast-aware content delivery and cooperative content sharing. In [27], Fang *et al.* investigated the economic interactions between Internet service providers and content providers in wireless networks, focusing on optimizing content delivery through edge caches and content popularity. They proposed a centralized model to solve the profit split problem and employed a Stackelberg game for a distributed solution.

Furthermore, there are already studies on collaborative caching based on MADRL. In [28], Araf *et al.* employed a multi-agent actor-critic reinforcement learning approach, combined with UAV (Unmanned Aerial Vehicle) assistance, to tackle the problem of optimizing cooperative caching strategies on the network edge. In [29], Chen *et al.* proposed a multi-agent actor-critic framework with variational recurrent neural networks (VRNN) to estimate content popularity. Through multi-agent deep reinforcement learning, the cooperative edge caching algorithm substantially improved the benefits of edge caches in ultra-dense next-generation networks. In [30], Jiang *et al.* employed multi-agent reinforcement learning to design content caching strategies in mobile device-to-device (D2D) networks, addressing the challenge of content caching without prior knowledge of popularity distribution. The proposed approach, using Q-learning, maximized caching rewards by treating users as agents and caching decisions as actions. In [31], Wu *et al.* proposed the integration of cooperative coded caching using maximum distance separable (MDS) codes in small cell networks to alleviate traffic loads. To adapt to dynamic and unknown content popularity, a multi-agent deep reinforcement learning framework was introduced to intelligently update cached content, with a focus on minimizing long-term expected front-haul traffic loads, resulting in improved caching efficiency. In [32], Zhong *et al.* investigated cache placement by using the collaborative MADRL algorithm in ultra-dense small cell networks (SCNs) to maximize cache hit rates in centralized and decentralized scenario settings. In [33], Jiang *et al.* explored edge caching for the internet of vehicles (IoVs) using distributed MARL. They introduced a hierarchical caching architecture to minimize content delivery overhead, extended single-agent reinforcement learning to a multi-agent system, and achieved superior performance in comparison to other caching strategies through simulation results. In [37], Song *et al.* addressed the content caching and sharing problem in cooperative base stations while considering unknown content popularity distribution and content retrieval costs. This paper formulated the problem as a multi-armed bandit (MAB) learning problem, jointly optimizing caching and sharing strategies. Two algorithms, a centralized approach using semidefinite relaxation and a decentralized approach based on the alternating direction method of multipliers (ADMM), were proposed to solve the problem efficiently. In [38], Cai *et al.* proposed a cross-tier cooperative caching architecture for all content that allows distributed cache nodes to cooperate, using a MADRL approach to model the decision process between heterogeneous cache nodes. In [39], Wang *et al.* introduced an intelligent edge caching framework named MacoCache, which incorporated MADRL to enhance edge caching performance in distributed environments. By employing the advanced actor-critic method and integrating long short-term memory (LSTM) for time series dynamics, MacoCache successfully reduced latency by an average of 21% and costs by 26% compared to existing learning-based caching solutions. In [40], Chen *et al.* utilized a refined MADRL tailored for the 5G network structure to improve scalable video coding (SVC)-based caching and video quality service strategies. The introduced value-decomposed dimensional network (VDDN) algorithm and a dimension decomposition method in the dueling deep Q-network addressed the challenges of large action spaces and computational efficiency. In [41], Zhang *et al.* introduced an edge caching model for IoV content distribution, allowing vehicles to collaboratively cache content and account for varying content popularity and channel conditions. An autonomous decision-making approach, an edge caching approach for the IoV based on multi-agent deep reinforcement learning, was proposed where vehicles act as agents to optimize caching decisions. However, most previous studies have primarily used MADRL to tackle issues in collaborative content caching. It is important to point out that these studies often did not address the more complex challenges associated with predicting what contents will be popular in the future. Specifically, they did not focus on the intricacies of using machine learning to accurately forecast content popularity, which is crucial for effective content management.

Recently, there have been several works on designing content caching schemes based on MADRL and FL. In [42], Chang *et al.* introduced a federated multi-agent reinforcement

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2024.3403842

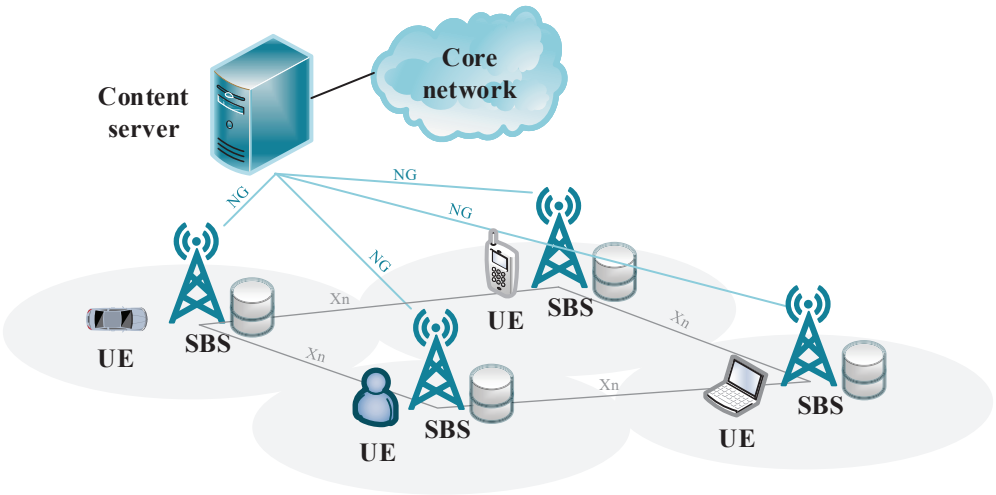


Fig. 1: Cooperative edge caching system

Learning (FMRL) method for optimizing edge caching. Using a Markov decision process (MDP) and a specialized regression model, it predicted content popularity, improved server rewards, and optimized cache decision-making. An enhanced FL technique aggregated local models, improving training and preserving user privacy. In [43], Prathiba *et al.* introduced the federated Learning and edge cache-assisted Cybertwin framework in response to rising autonomous vehicle (AV) technology and mobile traffic. This framework integrated the cybertwin concept to merge physical and digital systems. It employed the federated MADRL algorithm for optimized learning, considering various factors. Additionally, the federated reinforcement learning-based edge caching algorithm was used for efficient caching in 6G vehicle-to-everything networks. In [44], Sun *et al.* employed FL to perceive user preferences across various districts without exposing user requests. A content blocks-based RL is utilized for intelligent caching, taking into account the receiving scenarios of AVs. By integrating FL with MARL, this paper achieved distributed training of local models, which not only reduced the state-action space dimension but also circumvented local optimal solutions. However, the previous studies did not take into account the personalization needs of individual UE in FL training. Additionally, they overlooked the fact that different SBSs may have various popular contents, which can significantly impact caching strategies.

In conclusion, previous research has not adequately addressed two key aspects in designing content caching schemes. Firstly, FL approaches have not considered the personalized and elastic requirements of each UE, which is essential for optimizing user-specific performance. Secondly, MADRL methods have largely ignored the importance of predicting popular contents before making collaborative caching decisions. These gaps in the literature have motivated our research, as we aim to develop personalized models for UEs while also focusing on proactive collaborative caching strategies based on predicted content popularity.

III. System Model

As illustrated in Fig. 1, we consider a cooperative edge caching system in the next-generation network consisting of a content server (CS), a set of SBSs denoted as $\mathbb{B} = \{1, \ldots, b, \ldots, B\}$, where B is the number of SBSs, and a certain number of UEs. Each UE fetches the requested contents from SBSs or CS for multiple users, and stores a large amount of users' data, i.e., local data. Each data is a representation in vector form that encompasses various aspects of users' information such as the users' identification number (ID), gender, age, the contents that users may request, and users' ratings for the contents that they have previously requested. The rating is between 0 and 1 which can measure a user's interest in previously requested content, and this rating method is common in many research and application scenarios [47], [50]. 0 represents that the user is not interested in this requested content or the user has not requested this content before, while 1 represents that the user is most interested in this requested content. Each UE randomly selects a portion of local data to establish a training set, and the remaining data is utilized as a test set. The CS is linked to the core network by a backhaul link, and it stores all contents that users can request. Each SBS is connected to the CS through a backhaul link connected by the next-generation (NG) interface. SBSs are deployed at the network edge close to the UEs. The time duration is divided into time slots. In each time slot, each UE collects contents that users want to request and generates a requested information. This requested information may be a small-sized list containing the contents labels which users desire. Therefore, the requested information will not occupy too much resources. Then UEs send these requested information to their local SBSs to fetch the requested contents. The cache capacity of a single SBS is actually very limited, the adjacent SBSs can share cached contents with each other through a wired-link connected by the Xn interface. Hence, a UE can fetch a requested content from the local SBS, adjacent SBSs or CS. Specifically,

1) Local SBS: If the requested content is cached in the local SBS, the UE can fetch the requested content directly from the local SBS.

TABLE I: Main notations.

Notation	Description	Notation	Description
$\boldsymbol{a^t}$	the global action at time slot t.	a_b^t	the action of SBS b at time slot t.
B	the number of SBSs.	b_d	the bias vector of the decoder network.
b_e	the bias vector of the encoder network.	C	the cache capacity of each SBS.
c_b^t	the current contents cached by SBS b at time slot t.	d	the size of entire data of all UEs within the coverage of SBS.
$d(\cdot)$	the non-linear and logically-activated function of the discriminator network.	d_i	the size of the training data of each UE.
D	the replay buffer.	$D(z)$	the confidence of z.
e	the number of iteration of the local AAE training for UE.	E'	the number of episodes in the testing phase of MADRL algorithm.
F_p	the number of predicted popular contents.	$J(\boldsymbol{\pi^t})$	the expected long-term discounted reward at time slot t.
$\|L\|$	the number of layers of the AAE model.	M	the number of transition tuples in a mini-batch.
$n_{i,k}^r$	local training data set of UE i in the iteration k of round k.	$p(\cdot)$	the non-linear and logically-activated function of the decoder network.
$P(z)$	the predefined distribution.	p_b	the predicted popular contents of SBS b coverage.
$q(\cdot)$	the non-linear and logically-activated function of the encoder network.	R^t	the global reward at the time slot t.
R_b^t	the local reward of SBS b at time slot t.	R_{max}	the number of rounds to train the AAE model in the elastic FL.
$r_b^r(t)$	the number of contents that SBS b fetches from the local cache at time slot t.	$r_{b,c}^r(t)$	the number of contents that SBS b fetches from CS at time slot t.
$r_{b,e}^r(t)$	the contents replaced in SBS b during time slot t.	$r_{b,n}^r(t)$	the number of contents that SBS b fetches from adjacent SBSs at time slot t.
s_b^t	the local state of SBS b at time slot t.	$\boldsymbol{s^t}$	the global state at time slot t.
T	the number of time slots of the MADRL algorithm.	X_i^r	the rating matrix of UE i in round r.
$\tilde{X}_i^r$	the reconstructed rating matrix of UE i in round r.	α	the cost of fetching the contents from local SBS b.
α_i	the elastic parameter of UE i.	β	the cost that the UE within the coverage of local SBS b fetch the contents from adjacent SBSs.
χ	the cost of fetching the contents from CS.	δ	the cost of replacing a content.
γ	the discount factor.	η	the fixed learning rate.
θ_b	the parameters of the actor network of agent b.	θ_b'	the parameters of the target critic network of agent b.
φ	the parameters of the global critic network.	φ'	the parameters of the target global critic network.
τ	the constant is used to update the target networks.	ω^r	the global AAE model in round r.
ω_e^r	the global encoder network in round r.	ω_d^r	the global decoder network in round r.
$\omega_{d'}^r$	the global discriminator network in round r.	ω_i^r	the local model of UE i in round r.

2) Adjacent SBS: If the requested content is not cached in the local SBS but an adjacent SBS caches the request content, the local SBS can fetch the requested content from the adjacent SBS and then deliver it to the UE. If multiple adjacent SBSs have all cached the requested content, the local SBS randomly selects one of the adjacent SBSs to fetch the requested content.

3) CS: If both the local SBS and adjacent SBSs do not cache the requested content, the local SBS fetches the requested content from the CS and then delivers it to the UE.

In order to enable UEs within the coverage of each SBS to fetch requested contents more effectively, each SBS needs to accurately predict the popular contents. These popular contents are predicted based on users' ratings for the contents. However, it is difficult to distinguish whether the user is interested in this content or has requested it before if a rating for one content is 0. Therefore, it is necessary to predict and reconstruct the ratings of each content. The AAE model can effectively extract hidden features of ratings and reconstruct the ratings for each content. Hence the reconstructed ratings will contain fewer 0. This helps to predict the popular contents more accurately. Thus the AAE model is adopted as the local model and global model in the FL algorithm and the popular contents for each SBS are predicted based on the trained AAE model. The proposed CEFMR scheme is described as follows. We first propose an elastic FL algorithm which can assign specific weight for each local model based on the differences between the global model and local model to train the personalized local model for each UE. Then we propose a popular content prediction algorithm to predict the popular contents for each SBS based on the trained AAE model. Finally, due to the limited cache capacity of each SBS, we construct a MADRL framework and then adopt the MADDPG algorithm to decide where the predicted popular contents are collaboratively cached among SBSs to reduce the cost for fetching contents. We will elaborate the CEFMR scheme in the next section. The main notations are listed in Table I.

IV. Cooperative Edge Caching Scheme

In this section, we will introduce the CEFMR cooperative edge caching scheme. We first propose an elastic FL algorithm to train the personalized AAE model for each UE, then a popular content prediction algorithm is proposed to predict the popular contents for each SBS based on the trained AAE model. Finally, we construct a MADRL framework and then adopt the MADDPG algorithm to decide where the predicted popular contents are collaboratively cached among SBSs.

A. Elastic FL Algorithm

As illustrated in Fig. 2, elastic FL algorithm is executed through R_{max} rounds iteratively [48], [49]. Within each round,

the elastic FL algorithm consists of the following four steps.

1) Download Model: Each local SBS first generates the global AAE model in this step. Let ω^r be the parameters of the global AAE model in round r. The global AAE model consists of the AEN and GAN. Fig. 3 illustrates the framework of the AAE model. The upper layer of the AAE is an AEN which includes an encoder network and a decoder network. The bottom layer is a GAN which consists of a generator network and a discriminator network, where the encoder network of the AEN acts as the generator network. Let ω_e^r, ω_d^r and $\omega_{d'}^r$ be the parameters of the encoder network, decoder network and discriminator network in the round r, respectively, thus $\omega^r = \{\omega_e^r, \omega_d^r, \omega_{d'}^r\}$. For the first round, the local SBS initializes its own global AAE model ω^0 and for the subsequent round r the local SBS updates the global AAE model ω^r at the end of round $r-1$.

2) Local Training: Next each UE i in the coverage of local SBS downloads the global AAE model ω^r and updates the model based on its local data. If the UE directly uses the global model ω^r as the initial model for updating its own local model, it would result in the waste of the local model trained in the last round and the loss of individual UE features. Hence, we propose the elastic FL algorithm which takes into account both the local model and global model. Specifically, similar to [15], we first define the following weight distance formula to focus on the correlation between the local model of UE i trained in round $r-1$, i.e., ω_i^{r-1}, and ω^r, i.e.,

$$\operatorname{dis}\left(\omega_i^{r-1}, \omega^r\right) = \frac{\left\|\omega_i^{r-1} - \omega^r\right\|}{\left\|\omega^r\right\|}. \tag{1}$$

As the AAE model consists of multiple layers of neural network, we introduce α_i^r as an elastic parameter, which denotes the gap between ω_i^{r-1} and ω^r of all layers in round r. Thus α_i^r is calculated as follows

$$\alpha_i^r = \frac{1}{|L| n_b} \sum_{l \in L} \operatorname{dis}\left(\omega_{i,l}^{r-1}, \omega_l^r\right), \tag{2}$$

where $\omega_{i,l}^{r-1}$ is the l-th layer of ω_i^{r-1} and ω_l^r is the l-th layer of ω^r. L represents all the layers of the AAE model and $|L|$ denotes the number of layers. n_b is the number of the UE in the SBS b. Then the initial local model of UE i in round r is calculated as

$$\omega_i^r \leftarrow \alpha_i^r \omega^r + \left(1 - \alpha_i^r\right) \omega_i^{r-1}, \tag{3}$$

where α_i^r dynamically adjusts the influence of the global model on the local model, allowing the local model to have personalized features.

Then each UE updates ω_i^r iteratively through local training based on its local training data. During each iteration k, UE i conducts a random sampling process to select training data $n_{i,k}^r$ from the overall training set. Subsequently, the training data $n_{i,k}^r$ is utilized to train the local AAE model of UE i. The training process consists of the reconstruction stage and the adversarial regularization stage. The AAE model is trained alternatively between these two stages.

The reconstruction stage which mainly focuses on training the AEN is first executed. Specifically, the encoder network first takes each original training data x from $n_{i,k}^r$ and transforms x by mapping x to the network. This process yields the hidden feature representation of x, denoted as $z(x)$. The transformation is expressed as $z(x) = q\left(W_{i,e,k}^r x + b_{i,e,k}^r\right)$, here $W_{i,e,k}^r$ and $b_{i,e,k}^r$ are the weight matrix and bias vector of the encoder network of UE i in iteration k of round r, respectively. To make the paper clearer, we simplify $W_{i,e,k}^r$ and $b_{i,e,k}^r$ to $W_{e,k}$ and $b_{e,k}$, respectively. We also apply similar simplifications to symbols that will appear later.

Then the decoder network calculates the reconstructed input $\tilde{x}$, i.e., $\tilde{x} = p\left(W_{d,k} z + b_{d,k}\right)$, where $W_{d,k}$ and $b_{d,k}$ are the weight matrix and bias vector of the decoder network in iteration k, respectively. $p(\cdot)$ and $q(\cdot)$ are non-linear and logically-activated functions. Afterwards, the local loss function of each data x for the encoder network and the decoder network is calculated as

$$l(\omega_{e,k}, \omega_{d,k}; x) = (x - \tilde{x})^2, \tag{4}$$

where $\omega_{e,k}$ represents the parameters of the encoder network in iteration k, which includes $W_{e,k}$ and $b_{e,k}$, i.e., $\omega_{e,k} = \{W_{e,k}, b_{e,k}\}$. $\omega_{d,k}$ represents the parameters of the decoder network in iteration k, which includes $W_{d,k}$ and $b_{d,k}$, i.e., $\omega_{d,k} = \{W_{d,k}, b_{d,k}\}$. Then the loss function of all the data in $n_{i,k}^r$ for the encoder network and decoder network is calculated as

$$f(\omega_{e,k}, \omega_{d,k}) = \frac{1}{d_i} \sum_{x \in n_{i,k}^r} l(\omega_{e,k}, \omega_{d,k}; x), \tag{5}$$

where d_i is the size of $n_{i,k}^r$. Let $\nabla f\left(\omega_{e,k}, \omega_{d,k}\right)$ be the gradient of $f(\omega_{e,k}, \omega_{d,k})$. Then the parameters of the encoder network and decoder network in the reconstruction stage are updated respectively as

$$\omega_{e,k} \leftarrow \omega_{e,k} - \eta \nabla f\left(\omega_{e,k}, \omega_{d,k}\right), \tag{6}$$

$$\omega_{d,k+1} \leftarrow \omega_{d,k} - \eta \nabla f_k\left(\omega_{e,k}, \omega_{d,k}\right), \tag{7}$$

where η is the local learning rate. The reconstruction stage is finished for iteration k. It's important to note that the encoder network will be updated again in the subsequent adversarial regularization stage. Therefore, in Eq. (6) we still adopt the notation $\omega_{e,k}$ to represent the updated parameters of the encoder network after the reconstruction stage.

Afterwards, the adversarial regularization stage is executed, where the generator network and discriminator network will be alternately optimized. Specifically, the training of the discriminator network is first performed. The encoder network first maps x to obtain a vector, i.e., $z'(x) = q\left(W_{e,k} x + b_{e,k}\right)$. It is important to note that $W_{e,k}$ and $b_{e,k}$ have been updated in the previous reconstruction stage. Then $z'(x)$ will be input into the discriminator network to get its confidence $D(z'(x))$, i.e., $D(z'(x)) = d\left(W_{d',k} x + b_{d',k}\right)$, where $W_{d',k}$ and $b_{d',k}$ are the weight matrix and bias vector of the discriminator network in iteration k, respectively. $d(\cdot)$ is also the non-linear and logically-activated function. At the same time, a vector $\tilde{z}$ is sampled from a simple predefined distribution $P(z)$. Similarly, $\tilde{z}$ is also input to the discriminator network to obtain its confidence $D(\tilde{z})$. The objective of the discriminator network is to maximize its ability to correctly distinguish $z'(x)$ and

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2024.3403842

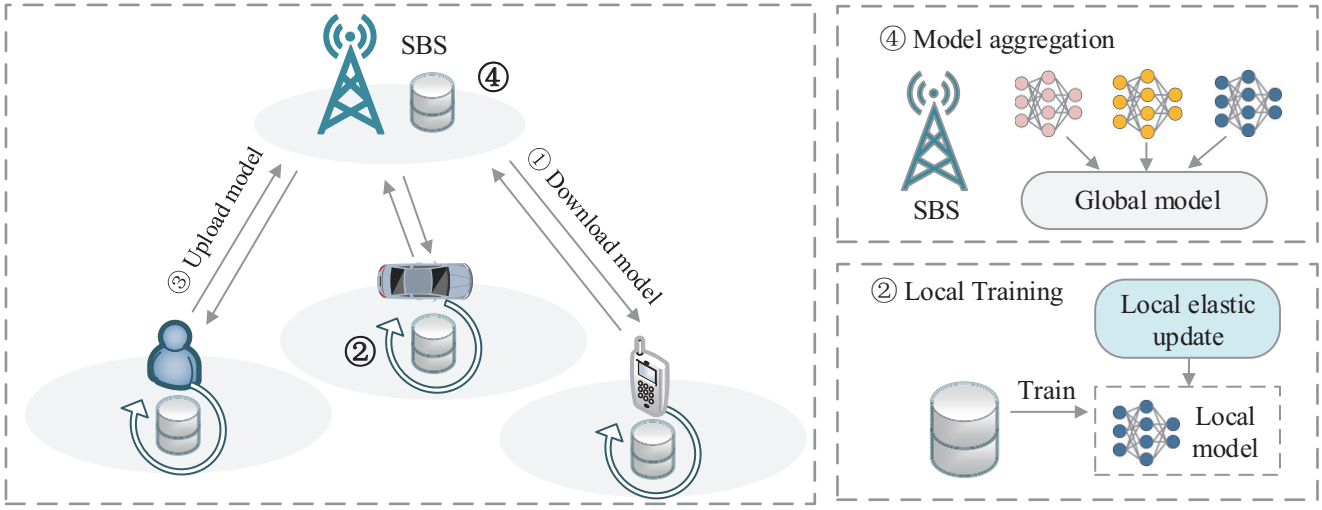


Fig. 2: Elastic federated learning

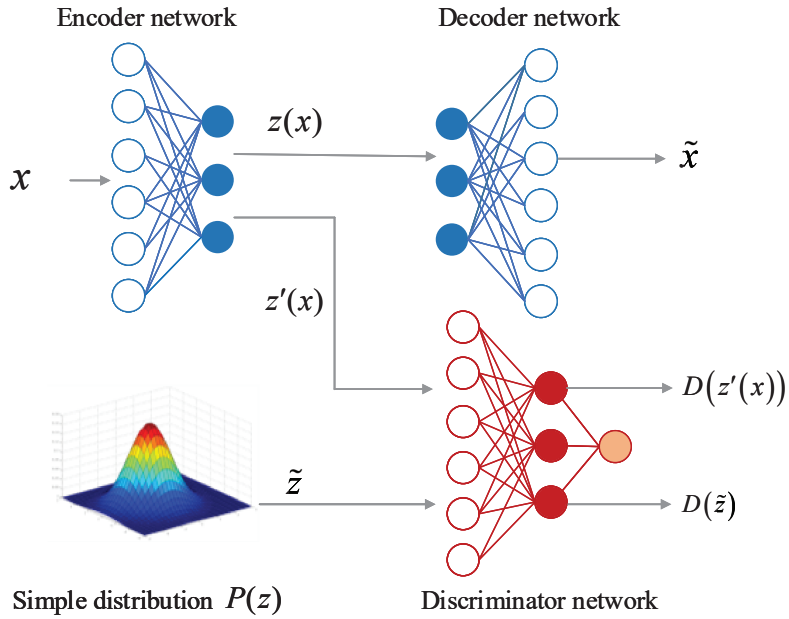


Fig. 3: AAE model

$\tilde{z}$. Therefore, the loss function of data x for the discriminator network can be calculated as,

$$L(\omega_{d',k};x) = -E[logD(\tilde{z})] - E[log(1 - D(z'(x)))], \quad (8)$$

where $\omega_{d',k}$ represents the parameters of the discriminator network in iteration k, which includes $W_{d',k}$ and $b_{d',k}$, i.e., $\omega_{d',k} = \{W_{d',k}, b_{d',k}\}$. Thus the loss function of all the data in $n^r_{i,k}$ for discriminator network is calculated as

$$F(\omega_{d',k}) = \frac{1}{d_i} \sum_{x \in n^r_{i,k}} L(\omega_{d',k};x). \quad (9)$$

Let $\nabla F(\omega_{d',k})$ be the gradient of $F(\omega_{d',k})$. Then the parameters of the discriminator network are updated as

$$\omega_{d',k+1} \leftarrow \omega_{d',k} - \eta \nabla F_{d',k}(\omega_{d',k}). \quad (10)$$

Then the parameters of encoder network is updated. The loss function of data x for the encoder network is defined as

$$L(\omega_{e,k};x) = -E[log(D(z'(x)))], \quad (11)$$

The loss function of all the data for the encoder network is calculated as

$$F(\omega_{e,k}) = \frac{1}{d_i} \sum_{x \in n^r_{i,k}} L(\omega_{e,k};x). \quad (12)$$

Let $\nabla F(\omega_{e,k})$ be the gradient of $F(\omega_{e,k})$. Then the parameters of the encoder network are updated as

$$\omega_{e,k+1} \leftarrow \omega_{e,k} - \eta \nabla F(\omega_{e,k}). \quad (13)$$

Iteration k is finished and UE i randomly samples some training data again to start the next iteration. When the number of iterations reaches the threshold e, UE i completes the local training.

Algorithm 1: The Elastic FL Algorithm

Initialize ω^r;
for *round* r *from* 1 *to* R_{max} **do**
  for *UE* $i = 1, 2, ...$ *in parallel* **do**
    Download global AAE model ω^r;
    $\omega_i^r \leftarrow$ **Local Updates**(ω^r, i);
    Upload the local AAE model ω_i^r to SBS;
  Calculate ω^{r+1} according to Eq. (15);
  return ω_{r+1}.
Local Update(ω, i):
Input: ω^r
Calculate the α_i according to Eq. (2);
Update the local AAE model ω_i^r according to Eq. (3);
for *each iteration* k *from* 1 *to* e **do**
  Randomly samples data $n_{i,k}^r$ from the training set;
  for *data* $x \in n_{i,k}^r$ **do**
    Calculate the loss function of data x for the encoder network and the decoder network according to Eq. (4);
  Calculated the local loss function of all the data for the encoder network and the decoder network in iteration k according to Eq. (5);
  Update the encoder network and the decoder network according to Eq. (6) and Eq. (7);
  for *data* $x \in n_{i,k}^r$ **do**
    Calculate the loss function of data x for the discriminator network according to Eq. (8);
  Calculated the local loss function of all the data for the discriminator network in iteration k according to Eq. (9);
  Update the discriminator network according to Eq. (10);
  for *data* $x \in n_{i,k}^r$ **do**
    Calculate the loss function of data x for the encoder network according to Eq. (11);
  Calculated the local loss function of all the data for the encoder network in iteration k according to Eq. (12);
  Update the encoder network according to Eq. (13);
return ω_i^r.

3) Upload model: Once each UE i completes the local training, then it uploads the updated local model ω_i^r to the local SBS, where ω_i^r consists of $\omega_{i,e}^r$, $\omega_{i,d}^r$ and $\omega_{i,d'}^r$, i.e., $\omega_i^r = \left\{\omega_{i,e}^r, \omega_{i,d}^r, \omega_{i,d'}^r\right\}$, here $\omega_{i,e}^r$, $\omega_{i,d}^r$ and $\omega_{i,d'}^r$ be the parameters of the encoder network, decoder network and discriminator network of UE i in the round r, respectively.

4) Weight aggregation: After all UEs within the coverage of the SBS upload their AAE models, the SBS generates a new global AAE model ω^{r+1} by calculating a weighted sum of all received local AAE models,

$$\omega^{r+1} = \omega^r - \eta \sum_{i=1} \frac{d_i}{d} \omega_i^r, \tag{14}$$

where d is the size of the total data for all UEs within the SBS coverage. So far, the elastic FL training for round r has been completed and the SBS has obtained an updated global AAE model ω^{r+1}, which is used for the training in the next round. When the number of rounds reaches R_{max}, the elastic FL training is finished and each SBS obtains the trained AAE model. For ease of understanding, Algorithm 1 illustrates the process of the elastic FL algorithm. Then each SBS sends the trained AAE model to all UEs in its coverage for content popularity prediction.

B. Popular Content Prediction

Next we propose a popular content prediction algorithm to predict the popular contents for each SBS based on the trained AAE model.

1) Data Preprocessing: UE i abstracts a rating matrix $\boldsymbol{X}_i^r$ from the data in the test set. The matrix's first dimension represents users' IDs, while the second dimension is users' ratings for all contents. Subsequently, each rating matrix $\boldsymbol{X}_i^r$ is fed into the updated AAE model, resulting in a reconstructed rating matrix $\tilde{\boldsymbol{X}}_i^r$. This reconstructed matrix is capable of extracting the hidden characteristics of the data and can be employed to approximate $\boldsymbol{X}_i^r$. Then, each UE also abstracts a matrix of personal information from the test set. This matrix has the first dimension representing user IDs, and the second dimension corresponding to the personal information of the users.

2) Cosine Similarity: Each UE i calculates the count of non-zero ratings for each user i in the matrix $\boldsymbol{X}_i^r$ and designates the users with the top $1/m$ numbers as active users. Following this, each UE merges the matrix $\tilde{\boldsymbol{X}}_i^r$ and the personal information matrix (the resultant matrix is denoted as $\boldsymbol{H}_i^r$) to determine the similarity between each active user and the rest of the users. The similarity between an active user u and another user v of UE i is calculated according to the cosine similarity [51]

$$\begin{aligned} \mathrm{sim}_{u,v}^{r,i} &= \cos\left(\boldsymbol{H}_i^r(u,:), \boldsymbol{H}_i^r(v,:)\right) \\ &= \frac{\boldsymbol{H}_i^r(u,:) \cdot \boldsymbol{H}_i^r(v,:)^T}{\left\|\boldsymbol{H}_i^r(u,:)\right\|_2 \times \left\|\boldsymbol{H}_i^r(v,:)\right\|_2}, \end{aligned} \tag{15}$$

where $\boldsymbol{H}_i^r(u,:)$ and $\boldsymbol{H}_i^r(v,:)$ represent the vectors corresponding to active user u and user v in the merged matrix, respectively. The terms $\left\|\boldsymbol{H}_i^r(u,:)\right\|_2$ and $\left\|\boldsymbol{H}_i^r(v,:)\right\|_2$ stand for the 2-norm (Euclidean length) of the vectors $\boldsymbol{H}_i^r(u,:)$ and $\boldsymbol{H}_i^r(b,:)$, respectively. For each active user u, UE i selects the users with the K highest similarities as the K nearest neighbors of user u. The ratings of these K nearest users can be interpreted as indicative of the preferences of user u.

3) Interested Contents: After determining the nearest users of active users in the matrix $\boldsymbol{X}_i^r$, the vectors of nearest users for each active user are extracted to create a matrix $\boldsymbol{H}_K$. In this matrix, the first dimension corresponds to the IDs of the nearest users for active users, while the second dimension signifies the ratings for content from these nearest users. In the matrix $\boldsymbol{H}_K$, any content with a non-zero rating from a user is considered as the interested content of that user. The number of such interested contents is then counted for each

user, and this count for a specific content is referred to as the popularity of the content. Each UE i selects the contents with the top F_p content popularities as the predicted interested contents and uploads these to the local SBS.

4) Popular Contents: After all UEs within the coverage range of the local SBS b upload their predicted interesting contents, SBS b collects and compares them, and selects F_p most interesting contents as the predicted popular contents p_b and then adopts MADDPG to cache C contents from p_b.

C. Cooperative Edge Caching Scheme Based on MADRL

Since each SBS has limited caching capacity, it is crucial for SBSs to collaborate in caching the predicted popular contents in order to reduce the cost for fetching contents. Therefore, we proposed the MADDPG algorithm for cooperative edge caching among SBSs, deciding SBS to cache suitable predicted popular contents, thus the cost can be optimized. Next, the MADRL framework is first formulated, which is the basis of the MADDPG algorithm. Then, the MADDPG algorithm will be introduced.

1) MADRL Framework: The MADRL framework includes agents, states, actions, rewards and policies [34]–[36]. In our cooperative edge caching, each SBS b can be viewed as an agent and makes its action based on the its local state. Then we will define states, actions, rewards and policies at time slot t as follows.

a) States: At time slot t, the local state of SBS b is defined as

$$\boldsymbol{s_b^t} = \left\{c_b^t, p_b^t\right\}, \tag{16}$$

where p_b^t represents the predicted popular contents within the coverage area of SBS b, c_b^t represents the current contents cached by SBS b. It should be noted that the size of p_b^t is F_p while the size of c_b^t is C, and $C < F_p$. The global state is defined as

$$\boldsymbol{s^t} = \left\{s_1^t, \ldots, s_b^t, \ldots, s_B^t\right\}. \tag{17}$$

b) Actions: Once SBS b predicts new popular contents p_b^t at time slot t, it needs to select an appropriate contents from p_b^t to replace part of the contents in c_b^t. Therefore, at time slot t, the action of SBS b is defined as

$$\boldsymbol{a_b^t} = \left\{a_{b,1}^t, \ldots, a_{b,f}^t, \ldots, a_{b,F_p}^t\right\}, \tag{18}$$

where each $a_{b,f}^t$ is a binary variable indicating if the f-th content in p_b is stored or not. 0 stands for that the f-th popular content is not stored in SBS b, while 1 indicates that it will be cached. Let $r_{b,e}^t$ be the number of 1 in $\boldsymbol{a_b^t}$, thus the number of 0 is $F_p - r_{b,e}^t$. The global action is defined as

$$\boldsymbol{a^t} = \left\{a_1^t, \ldots, a_b^t, \ldots, a_B^t\right\}. \tag{19}$$

c) Reward function: Compared to UEs fetching contents from CS, it can significantly reduce costs by using the cooperative edge caching system. Thus we define the reward as the saved cost. Next, we will first introduce the cost which UEs fetch contents from the cooperative edge caching system, then introduce the cost which UEs fetch all contents only from CS, and the difference between these two costs is the saved cost, which is our defined reward.

In the cooperative edge caching system, a UE can fetch requested contents directly from the local SBS, CS or adjacent SBSs. The three optional methods of obtaining the requested contents usually correspond to different costs.

The UE can fetch the requested contents from the local SBS b. Let α denote the cost of fetching the contents from local SBS b. Assume that UEs fetch r_b^t contents from local SBS b at the time slot t, thus the cost is αr_b^t.

The UE can indirectly fetch the requested contents from adjacent SBSs. Let β denote the cost that the UEs within the coverage of local SBS b fetch the contents from adjacent SBSs. Since the delivery of contents between SBSs will consume backhaul resources, the cost of fetching contents from adjacent SBSs is higher than that of fetching the contents from local SBS b, i.e., $\beta > \alpha$. Assume that the UEs in the coverage of local SBS b fetch $r_{b,n}^t$ contents from adjacent SBSs. In this case, the cost is $\beta r_{b,n}^t$.

The UE can fetch the requested content from CS. Let χ denote the cost of fetching the contents from CS. The cost of fetching contents from CS is much higher than that of fetching contents from SBSs owing to the resource consumption of backhaul and core network, i.e., $\chi > \beta$ and $\chi > \alpha$. Assuming that the UEs in the coverage of local SBS b fetch $r_{b,c}^t$ contents from CS at time slot t. In this case, the cost is $\chi r_{b,c}^t$.

In addition, the contents replaced by SBS b are all from CS, it will result in the additional costs as it requires the consumption of backhaul resources between SBS b and CS. Denote δ as the cost of replacing a content. The cost of replacing caching contents is $\delta r_{b,e}^t$. Thus, the total cost of SBS b is $\alpha r_b^t + \beta r_{b,n}^t + \chi r_{b,c}^t + \delta r_{b,e}^t$.

Noted that the constants α, β, χ and δ can be represented with actual physical significance, that is, these values can be explained as a general expression of the benefits from content caching, such as energy saving, expense reduction and so on [29], [37], [46].

Edge caching in the cooperative edge caching system can significantly reduce costs. Without edge caching, all UEs would have to fetch contents from CS, which leads to a great cost, i.e., $\chi\left(r_b^t + r_{b,n}^t + r_{b,c}^t\right)$, thus the reward of SBS b at time slot t can be defined as the saved cost, i.e.,

$$\begin{aligned} R_b^t &= \chi\left(r_b^t + r_{b,n}^t + r_{b,c}^t\right) \\ &- \left(\alpha r_b^t + \beta r_{b,n}^t + \chi r_{b,c}^t + \delta r_{b,e}^t\right) \\ &= (\chi - \alpha) r_b^t + (\chi - \beta) r_{b,n}^t - \delta r_{b,e}^t \end{aligned}. \tag{20}$$

Note that the higher the cost savings, i.e., the larger R_b^t, will cause more effective edge caching. We use $\boldsymbol{R_L^t} = [R_1^t, \ldots, R_b^t, \ldots, R_B^t]$ to denote all SBSs' reward. The global reward at time slot t is calculated as

$$R^t = \frac{1}{B}\sum_{b=1}^{B} R_b^t. \tag{21}$$

Let $\boldsymbol{\pi^t} = \{\pi_1^t, \ldots, \pi_b^t, \ldots, \pi_B^t\}$ denotes the caching policies at time slot t. $\boldsymbol{\pi^t}$ will map state $\boldsymbol{s^t}$ to action $\boldsymbol{a^t}$, i.e., $\boldsymbol{a^t} = \boldsymbol{\pi}(\boldsymbol{s^t})$. The expected long-term discounted reward of MADRL

is calculated as

$$J(\boldsymbol{\pi^t}) = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t R^t\right]. \tag{22}$$

where $\gamma \in (0,1)$ is the discount factor. Our cooperative edge caching problem can be formulated as a multi-agent decision problem to maximize the expected long-term discounted reward to get the optimal policies $\boldsymbol{\pi}^*$, which is calculated as

$$\boldsymbol{\pi}^* = \arg\min J(\boldsymbol{\pi}). \tag{23}$$

2) MADDPG algorithm: The MADDPG algorithm is proposed for the cooperative edge caching. Next we first introduce the training stage of MADDPG algorithm to obtain the optimal policies, then describe the testing stage to test performance based on the optimal policies.

The MADDPG algorithm is based on the multi-agent actor-critic framework [52]. Each agent b has a local actor network, a local critic network, a local target actor network and a local target critic network. The local actor network of agent b outputs the policy based on the local state, then the action can be obtained according to the policy. The local critic network of agent b is used to evaluate the action according to the local reward. Two local target networks can ensure the stability of the algorithm, whose network structure is same as that of local actor network and local critic network, respectively. All agents share two centralized global critic networks and two global target critic networks, all of them are located at CS. Two global critic networks provide a global perspective based on the global states and actions of all agents, enabling each agent to understand and adapt to the global environment. Two global target critic networks which have the same network structure as the global critic networks, respectively, are also used to ensure the stability.

Let θ_b and ϕ_b are the parameters of the local actor network and local critic network of agent b, respectively, while θ'_b and ϕ'_b are the parameters of the local target actor network and local target critic network of agent b, respectively. Let φ_1, φ_2, φ'_1 and φ'_2 be the parameters of two global critic networks and two global target critic networks, respectively. The detailed training stage for MADDPG algorithm is further introduced, and Algorithm 2 describes the specific process of MADDPG training stage.

a) Training Stage: Firstly, each agent b randomly initializes its local actor network parameters θ_b and local critic network parameters ϕ_b. CS also initializes the two global critic network parameters φ_1 and φ_2. The local target actor network parameters θ'_b and local target critic network parameters ϕ'_b of each agent b are initialized as θ_b and ϕ_b, while two target global critic network parameters φ'_1 and φ'_2 are initialized as φ_1 and φ_2. The replay buffer D which is located at CS is equipped with substantial storage capacity to store transition tuples in each time slot t, i.e., $\boldsymbol{s^t}$, $\boldsymbol{a^t}$, R^t, $\boldsymbol{R_L^t}$ and $\boldsymbol{s^{t+1}}$. Replay buffer D can facilitate the efficient sharing of experience across the agents.

Next, the algorithm executes for R_{max} episodes. In the first episode, each SBS b obtains F_p popular contents p_b^1 from predicted interested contents uploaded by UEs within its own coverage, and randomly selects C contents for caching. Thus each SBS b can obtain the initial caching contents c_b^1. Based on c_b^1 and p_b^1, each SBS b can obtain its initial local state $\boldsymbol{s_b^1}$ according to Eq. (16), and the initial global state can also be further obtained according to Eq. (17), i.e., $\boldsymbol{s^1} = \left[s_1^1, \ldots, s_b^1, \ldots, s_B^1\right]$.

Then algorithm is iteratively executed from slot 1 to slot T. In the first slot, each SBS b inputs its local state s_b^1 to its local actor network and gets its policy π_b^1, then the local action a_b^1 can be obtained based on π_b^1. Thus the global action can be obtained according to Eq. (19), i.e., $\boldsymbol{a^1} = \left[a_1^1, \ldots, a_b^1, \ldots, a_B^1\right]$. After each SBS b takes the local action a_b^1, it can get its local reward R_b^1 according to Eq. (20). Thus all agents' rewards can be obtained, i,e., $\boldsymbol{R_L^1} = \left[R_1^1, \ldots, R_b^1, \ldots, R_B^1\right]$. The global reward R^1 can also be obtained according to Eq. (21). In addition, each SBS b enters the next local state s_b^2 after taking the local action, and the next global state s^2 can be further

Algorithm 2: Training Stage for the MADDPG Framework

Input: γ, τ, θ_b, ϕ_b, φ_1, φ_2

Randomly initialize the θ_b, ϕ_b, φ_1, φ_2;
Initialize target networks by $\theta'_b \leftarrow \theta_b$, $\phi'_b \leftarrow \phi_b$, $\varphi'_1 \leftarrow \varphi_1$, $\varphi'_2 \leftarrow \varphi_2$;
Initialize replay buffer $\mathcal{D}$;
for *episode from* 1 *to* R_{max} **do**
  for *agent* b *from 1 to* B **do**
    Predict the popular contents p_b;
    Receive initial observation state s_b^1;
  $s^1 = \left[s_1^1, \ldots, s_b^1, \ldots, s_B^1\right]$;
  for *time slot* t *from* 1 *to* T **do**
    for *agent* b *from 1 to* B **do**
      Observe s_b^t and select action $a_b^t = \pi\left(s_b^t \mid \theta_b\right)$;
    $s^t = [s_1^t, \ldots, s_b^t, \ldots, s_B^t]$, $a^t = [a_1^t, \ldots, a_b^t, \ldots, a_B^t]$;
    Receive global and local rewards, R^t and R_L^t;
    Store transition $(s^t, a^t, R^t, R_L^t, s^{t+1})$ in ()();
    if *number of transition tuples in* $\mathcal{D}$ *is larger than* M **then**
      Randomly sample a mini-batch of M, $(s^i, a^i, R^i, R_L^i, s')$, from $\mathcal{D}$;
      Set $y_g^i = R^i + \gamma \min_{x=1,2} Q\left(s'^i, a'^i \mid \varphi'_x\right)$;
      Update global critics by minimizing the loss based on Eq. (25);
      Update target parameters based on Eq. (29);
      for *agent* b *from 1 to* B **do**
        Set $y_b^i = R_b^i + \gamma Q\left(s_b'^i, a_b'^i \mid \phi'_b\right)$;
        Update local critic network by minimizing the loss based on Eq. (27);
        Update the local actor network based on Eq. (28);
        Update target local networks according to Eqs. (30) and (31).

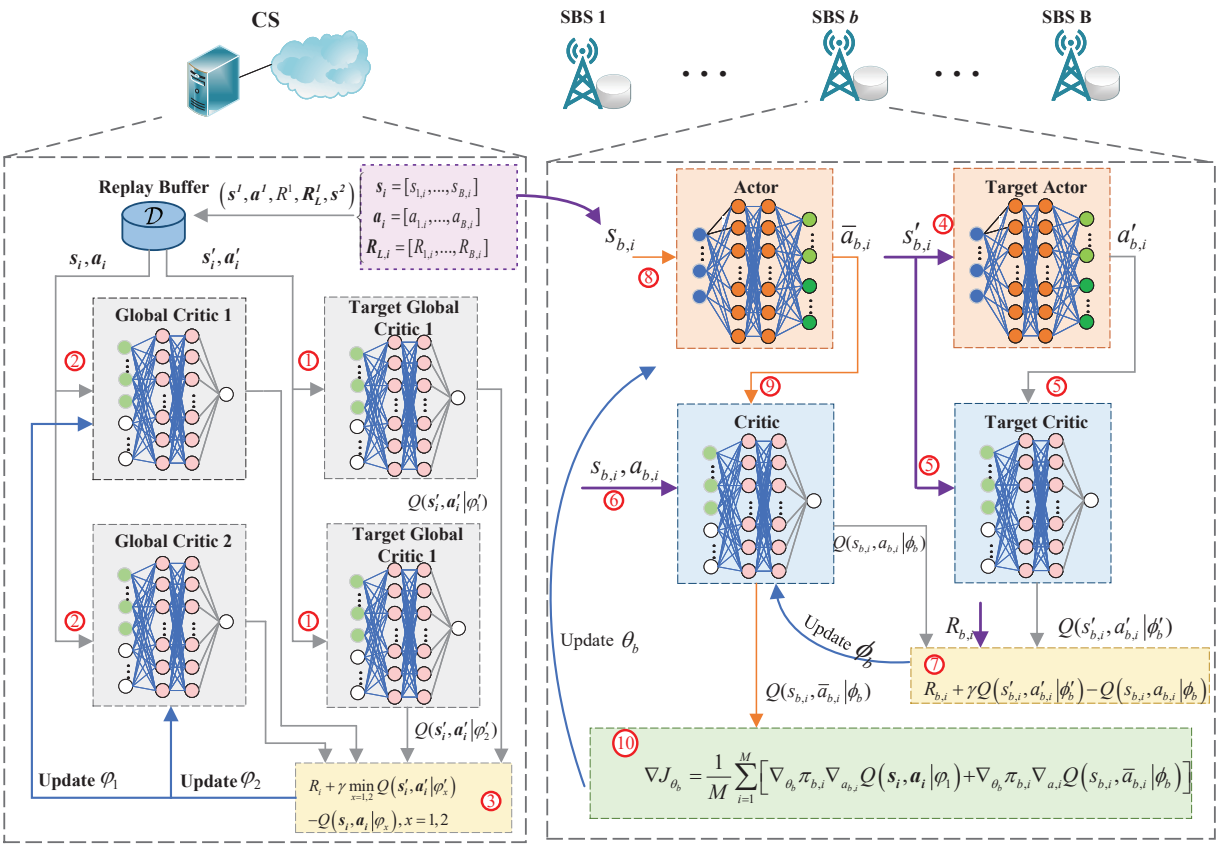


Fig. 4: MADDPG algorithm flow chart of the training stage

obtained, i.e., $\boldsymbol{s^2} = \left[s_1^2, \ldots, s_b^2, \ldots, s_B^2\right]$.

Then the transition tuple $\left(\boldsymbol{s^1}, \boldsymbol{a^1}, R^1, \boldsymbol{R_L^1}, \boldsymbol{s^2}\right)$ is stored in the replay buffer D. When the number of transition tuples stored in the replay buffer D is less than M, the algorithm starts the next slot.

When the number of stored transition tuples is greater than M in time slot t, CS first randomly samples M transition tuples from the replay buffer D to form a mini-batch. For the sake of simplicity, let $(\boldsymbol{s_i}, \boldsymbol{a_i}, R_i, \boldsymbol{R_{L,i}}, \boldsymbol{s_i'})\,(i = 1, 2, \ldots, M)$ represents the i-th transition tuple in the mini-batch. For the transition tuple i, SBS b first inputs its local state $s_{b,i}'$ into the local target local actor network and gets its policy $\pi_{b,i}'$, then the local action $a_{b,i}'$ can be obtained based on $\pi_{b,i}'$. All agents' actions $\boldsymbol{a_i'} = \left[a_{1,i}', \ldots, a_{b,i}', \ldots, a_{B,i}'\right]$ can be obtained. Then CS inputs $\boldsymbol{a_i'}$ and $\boldsymbol{s_i'}$ into two target global critic networks, and outputs two target action-state values $Q\left(\boldsymbol{s_i'}, \boldsymbol{a_i'} \mid \varphi_1'\right)$ and $Q\left(\boldsymbol{s_i'}, \boldsymbol{a_i'} \mid \varphi_2'\right)$, respectively. This process is shown as step 1 in Fig. 4. The target global value can be calculated as

$$y_{g,i} = R_i + \gamma \min_{x=1,2} Q\left(\boldsymbol{s_i'}, \boldsymbol{a_i'} \mid \varphi_x'\right). \tag{24}$$

Next CS inputs $\boldsymbol{a_i}$ and $\boldsymbol{s_i}$ into two global critic networks, and outputs two action-state values $Q\left(\boldsymbol{s_i}, \boldsymbol{a_i} \mid \varphi_1\right)$ and $Q\left(\boldsymbol{s_i}, \boldsymbol{a_i} \mid \varphi_2\right)$, respectively. This process is shown as step 2 in Fig. 4. The loss function of each global critic network is calculated as

$$L\left(\varphi_x\right) = \frac{1}{M} \sum_{i=1}^{M} \left[y_{g,i} - Q\left(\boldsymbol{s_i}, \boldsymbol{a_i} \mid \varphi_x\right)\right]^2, x = 1, 2, \tag{25}$$

The above equation is shown as step 3 in Fig. 4. The gradient of the each global critic network can be calculated as $\nabla L\left(\varphi_x\right), x = 1, 2$.

Then each SBS b trains its own local critic and local actor networks. Each SBS b first inputs its local state $s_{b,i}'$ into its local target actor network and obtains the local action $a_{b,i}'$. This process is shown as step 4 in Fig. 4. Next each SBS b inputs $s_{b,i}'$ and $a_{b,i}'$ into its local target critic network, and gets the local target action-state value $Q\left(s_{b,i}', a_{b,i}' \mid \phi_b'\right)$. This process is shown as step 5 in Fig. 4. Thus the target value of the local critic network can be calculated as

$$y_{b,i} = R_{b,i} + \gamma Q\left(s_{b,i}', a_{b,i}' \mid \phi_b'\right). \tag{26}$$

Then each SBS b inputs $s_{b,i}$ and $a_{b,i}$ into its local critic network, and gets the local action-state value $Q\left(s_{b,i}, a_{b,i} \mid \phi_b\right)$. This process is shown as step 6 in Fig. 4. The local critic loss function of SBS b can be calculated as

$$L\left(\phi_b\right) = \frac{1}{M} \sum_{i=1}^{M} \left[y_{b,i} - Q\left(s_{b,i}, a_{b,i} \mid \phi_b\right)\right]^2. \tag{27}$$

The above equation is shown as step 7 in Fig. 4. Then the gradient of the local critic network is calculated as $\nabla L\left(\phi_b\right)$.

Then each SBS b inputs $s_{b,i}$ into its local actor network and gets its policy $\bar{\pi}_{b,i}$, then the local action $\bar{a}_{b,i}$ can be obtained based on $\bar{\pi}_{b,i}$. Next each SBS b inputs $s_{b,i}$ and $\bar{a}_{b,i}$ into its local target critic network, and gets the local target action-state value $Q\left(s_{b,i}, \bar{a}_{b,i} \mid \phi_b\right)$. These processes is shown as steps 8 and 9 in Fig. 4. The gradient of the local actor network of SBS b is calculated as

$$\nabla J_{\theta_b} = \frac{1}{M}\sum_{i=1}^{M}\left[\nabla_{\theta_b}\pi_{b,i}\nabla_{a_{b,i}}Q\left(\boldsymbol{s_i}, \boldsymbol{a_i} \mid \varphi_1\right) + \nabla_{\theta_b}\pi_{b,i}\nabla_{a_{b,i}}Q\left(s_{b,i}, \bar{a}_{b,i} \mid \phi_b\right)\right]. \tag{28}$$

The above equation is shown as step 10 in Fig. 4. The first term in Eq. (28) is associated with the global critic network, similar with [52], we only use one action-state value from one of the global critic networks, i.e., $Q\left(\boldsymbol{s_i}, \boldsymbol{a_i} \mid \varphi_1\right)$. The second term corresponds to each agent's local critic network.

After that φ_1, φ_2, ϕ_b and θ_b are updated through gradient ascending based on $\nabla L\left(\varphi_1\right)$, $\nabla L\left(\varphi_2\right)$, $\nabla L\left(\phi_b\right)$ and ∇J_{θ_b}.

Then the parameters of the two target global critic networks, local actor network and the critic network are updated as

$$\varphi_x' = \tau\varphi_x + (1-\tau)\varphi_x', x = 1, 2, \tag{29}$$

$$\theta_b' = \tau\theta_b + (1-\tau)\theta_b', \tag{30}$$

$$\phi_b' = \tau\phi_b + (1-\tau)\phi_b'. \tag{31}$$

where $\tau \leq 1$ is a constant value. The update in this time slot is finished, and the above procedure is repeated for the next time slot. When the number of time slot reaches T, the current episode ends up. Then each SBS initialize its cached contents to start the next episode. When the number of episodes reaches N, the algorithm will terminate and we will get the optimal parameters θ_b^* of the local actor network of each SBS b. The flow chart of the MADDPG training stage is shown in Fig. 4.

b) Testing Stage: The testing stage omits the critic network, target actor network, target critic network of each SBS b, two global critic networks and two target global critic networks, as compared to the training stage. During the testing stage, the local state s_b^t of each SBS b in each time slot t is input into its local actor network with optimized parameters θ_b^*, and the optimal action which denotes the best caching placement can be obtained. The pseudo-code of the testing stage is shown in Algorithm 3.

D. Computational Complexity Analysis

In this section, we will analyze the computational complexity of our approach. The analysis focuses on the training stage due to the significant computation resource and time consumption in the training stage. Our methodology for analyzing the computational complexity is inspired by [53]. During a round, the training process includes two processes: the elastic FL training of AAE and the training of MADDPG. Because the training process requires significant computational resources to calculate gradients and update parameters, the computational complexity mainly consists of the complexity of computing gradients and the complexity of updating parameters.

Let G_E, G_D, $G_{D'}$, G_A, and G_C be the computational complexity of computing gradients for the encoder network, the decoder network, the discriminator network, the actor network, and the critic network, respectively. Let U_E, U_D, $U_{D'}$, U_A, and G_C be the computational complexity of updating parameters for the aforementioned networks. First, we analyze the complexity of the elastic FL training of AAE, which is influenced by the number of SBSs and the number of UEs covered by each SBS. Let n_b represent the number of UEs under SBS b, and each UE needs to undergo e iterations of local training. Therefore, the complexity of the elastic FL training of AAE is $O_{aarfl} = O\left(R_{max}\sum_{b=1}^{B} n_b e\left(2G_E + G_D + G_{D'} + 2U_E + U_D + U_{D'}\right)\right)$.

Next, we analyze the computational complexity of MADDPG training, which is mainly influenced by the number of SBSs. Since the local target actor and local actor have the same network structure, they have the same G_A and U_A. Moreover, each SBS's two local target critic networks, two local critic networks, two global critic networks, and two global target critic networks have the same network structure, so they are both G_C and U_C. It should be noted that the target networks do not need to caculate gradients, and the training and updating parameters process will not be activated until the tuples stored in the replay buffer are larger than M. Therefore, the computational complexity of MADDPG training is $O_{madrl} = O((R_{max}T - M)\left(B(G_A + U_E) + (2+B)(G_C + U_E)\right))$.

Thus, the total computational complexity is $O = O_{aarfl} + O_{madrl}$.

Algorithm 3: Testing Stage for the MADDPG Framework

for *episode from* 1 *to* E' **do**
  Predict the popular contents p_b^1;
  Receive initial local state s_b^1;
  for *time slot* t *from* 1 *to* T **do**
    for *agent* b *from 1 to* B **do**
      Generate caching placement based on the optimal policy $a_b^t = \pi\left(s_b^t \mid \theta_b^*\right)$;
      Execute action a_b^t, observe local reward R_b^t and obtain new local state s_b^{t+1};
    Observe global R^t and obtain next global state $\boldsymbol{s^{t+1}}$.

V. Simulation Results

In this section, simulation experiments are conducted to evaluate the performance of the proposed CEFMR scheme in the collaborative caching system.

TABLE II: Values of the parameters in the experiments.

Parameter	Value	Parameter	Value
R_{max}	1000	E'	100
M	256	T	100
α	1	β	30
χ	100	δ	100
γ	0.99	η	0.01
τ	0.001		

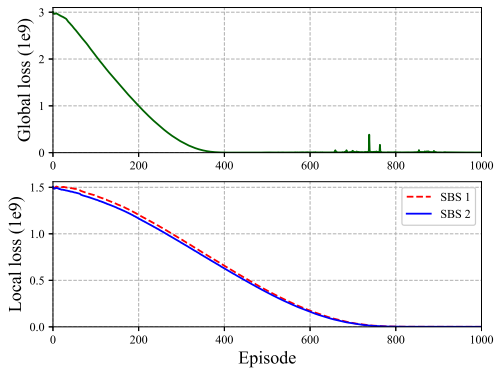


Fig. 5: Global and local losses during the training stage of MADDPG

A. Settings and Dataset

The simulation experiments in this paper are conducted based on Python 3.8 while leveraging the extensively acknowledged MovieLens 1M dataset, which encompasses nearly 100 million movie ratings given by 6040 anonymous users for 3883 movies [55]. A part of the movies that have been rated by users will be the requested contents for the UEs. Table II lists the values of all parameters in the simulation.

B. Training Stage

First, we present the training results of the MADDPG algorithm, where the number of SBSs is set to 2. Fig. 5 shows the average loss of the two global critic networks for the two SBSs, and two local losses of the local critic networks for each SBS. As can be seen from Fig. 5, the global loss decreases slowly until the $400th$ episode and then gradually stabilizes, while the local losses of two SBSs decrease slowly until the $750th$ episode and then gradually stabilize. The loss represents the error between the action-state value and the actual reward, thus the action-state value gradually approaches the actual reward with the progress of training the loss decreases. This means that MADDPG algorithm converges after training, and we can obtain an optimal local actor network for each SBS. It should be noted that the number of UEs will only affect the speed of elastic FL training and does not impact MADDPG. Additionally, the primary objective of this paper is to address the cooperative caching challenges among multiple MBSs. As a result, to maintain focus and ensure a controlled environment for our analysis, we set a fixed number of UEs during the training of elastic FL. Research on dynamic UE numbers has been explored in our previous work [51], [55].

C. Performance Evaluation

The experiments use the cost and cache hit ratio of the network as performance metrics. The cache hit ratio quantifies the utilization of the edge caching. If the content requested by a UE within the coverage of a local SBS b is cached in the SBS or adjacent SBSs, it is called a cache hit. Otherwise, it

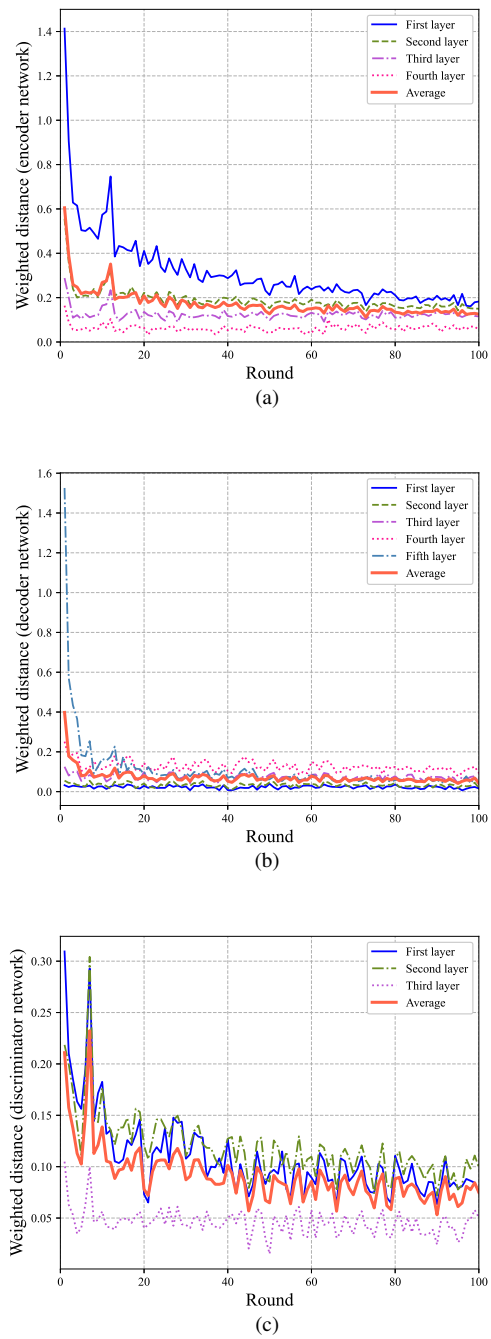


Fig. 6: Weight distance vs round. (a) Encoder network; (b) Decoder network; (c) Discriminator network.

is called a cache miss. Thus the cache ratio of SBS b can be calculated as

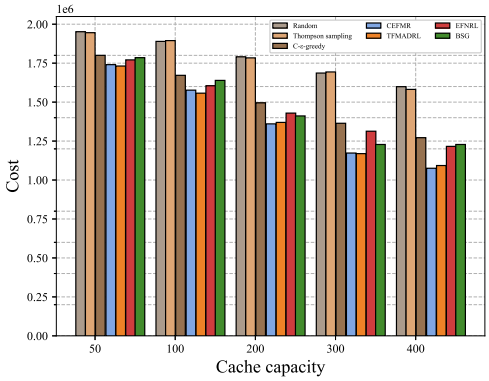
$$CH_b = \frac{\text{cache hits}}{\text{cache misses + cache hits}}. \tag{32}$$

The average cache hit ratio of all SBSs is calculated as

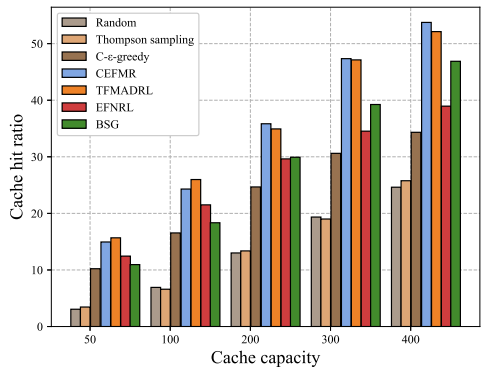
$$CH = \frac{1}{B}\sum_{b=1}^{B} CH_b. \tag{33}$$

In order to evaluate the performance of the proposed CEFMR scheme, three baseline caching schemes are intro-

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2024.3403842



(a)

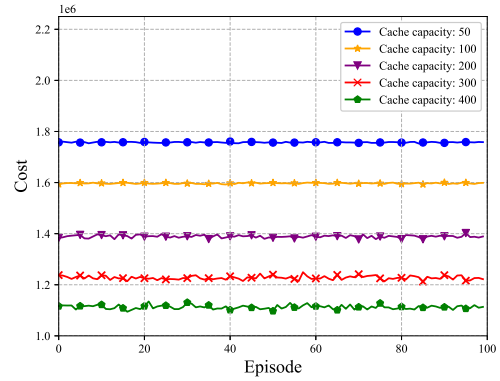


(b)

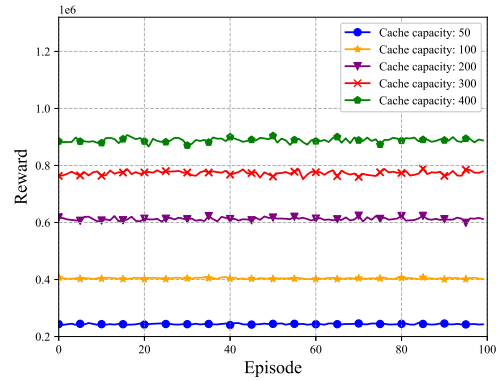
Fig. 7: Performance vs cache capacity under different schemes. (a) Cost; (b) Cache hit ratio.

duced as below for comparison.

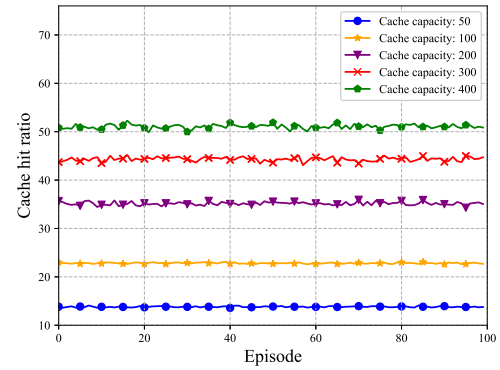
- Random: Each SBS randomly selects C contents from all contents for caching.
- C-ε-greedy: Each SBS caches the C contents with the highest number of requests based on the probability of $1-\varepsilon$, it randomly selects C contents for caching with a probability of ε. In the simulation, $\varepsilon = 0.1$.
- Thompson sampling: Each SBS updates the contents cached in that SBS based on the number of cache hits and cache misses in the previous time slot, and caches the C contents with the highest values.
- Bayes-Based Popularity Learning and Sequential Greedy Algorithm (BSG) [54]: The algorithm first initializes each content's popularity with a Beta distribution and updates their distribution based on the observed frequency of requests and non-requests for each content. Then the algorithm sorts SBS based on the sum of the popularity of UE requested contents within the SBS coverage, meanwhile it sorts all contents based on popularity. It then allocates most popular contents to the top-sorted SBS.
- Traditional Federated MADRL (TFMADRL): This scheme employs traditional federated learning to train the AAE model, and then utilizes MADDPG for collaborative caching.
- Elastic Federated learning with Non-RL (EFNRL): This scheme utilizes elastic federated learning to train the AAE model. During caching, each SBS sorts the predicted popularity of content and then caches the top C contents.



(a)



(b)



(c)

Fig. 8: Testing performance vs episode for different cache capacities. (a) Cost; (b) Reward; (c) Cache hit ratio.

For the Random scheme, the complexity is $O(C)$, where C is the cache capacity of the SBS. The C-ε-greedy scheme's complexity is $O(B \times m \log m)$, with B representing the number of UEs covered by each SBS, and m being the number of content requests by UEs. Thompson sampling has an approximate complexity of $O(e_t \times B \times m \log m)$. The BSG scheme's complexity is approximately $O(m_{all} \log m_{all})$, where m_{all} indicates all cached content in the CS. The complexity of both Traditional Federated MADRL and our CEFMR is approximated as $O_{aarfl} + O_{madrl}$. The complexity

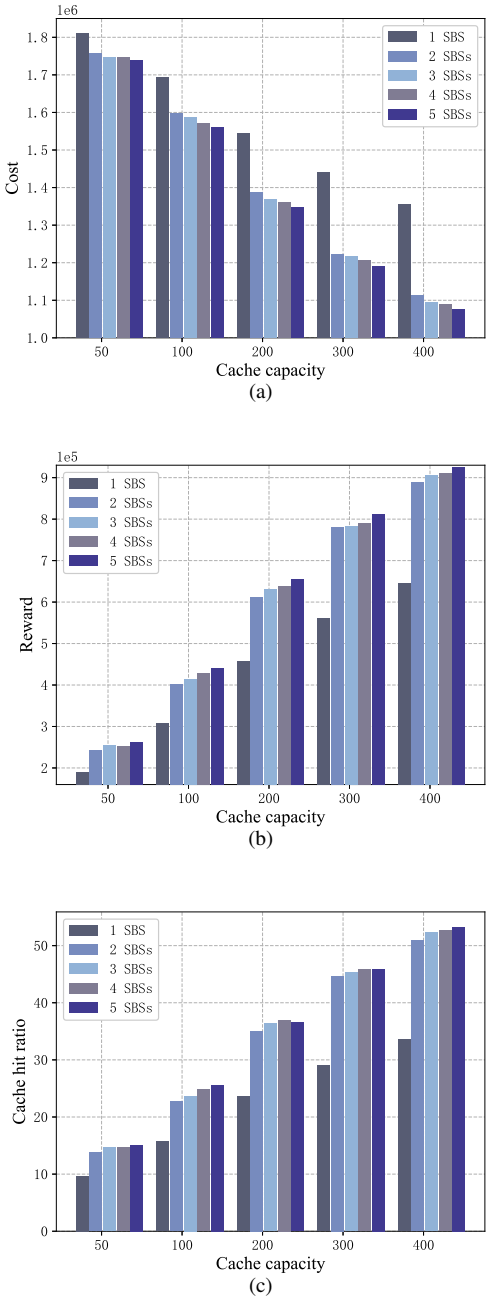


Fig. 9: Performance vs cache capacity for different number of SBSs. (a) Cost; (b) Reward; (c) Cache hit ratio.

of EFNRL is approximately O_{aarfl}.

Figs. 6-(a), (b) and (c) show the average weight distance of each layer between UEs' local AAE models and global AAE model, and the average weight distance of all layers between UEs' local AAE models and global AAE model under different rounds in elastic FL training. It can be seen that all the weight distances tend to decrease as round increases. After a certain number of rounds, all the weight distances converge. This is because as the training of elastic FL, both UEs' local models and global model gradually converge, and the local models are trained based on the downloaded global model, the divergence between local models and global model will gradually decrease. It also can be seen that all weight distances do not reduce to zero after convergence. This is because the data among UEs are diverse, thus the models trained locally by UEs maintain their own characteristics.

Figs. 7-(a) and (b) show the total cost of all SBSs, where the cost of each SBS b is $\alpha r_b^t + \beta r_{b,n}^t + \chi r_{b,c}^t + \delta r_{b,e}^t$, and cache hit ratio with different cache capacities of each SBS under different schemes, respectively. It can be seen that the costs of all schemes decrease as the cache capacity increases, and the cache hit ratios increase as the cache capacity increases. This is because all SBSs which have larger caching capacities can cache more contents. As a result, UEs can more easily fetch their requested contents from SBSs. This can alleviate the pressure on the CS and improve the caching performance of the system. The Random and Thompson Sampling schemes have poor performance. This is because these schemes do not cache contents based on predicting content popularity, while other five schemes determine the cached contents based on the predicted interested contents uploaded by the UEs within the coverage of SBS. Both CEFMR and TFMADRL schemes outperform EFNRL, BSG, and C-ε-greedy schemes. This is because CEFMR and TFMADRL schemes employ cooperative caching using the MADDPG method. CEFMR and TFMADRL schemes also outperform BSG scheme because they utilize a more efficient AAE model for popularity prediction. Furthermore, from the figure, it can be observed that as cache capacity increases, CEFMR scheme gradually outperforms TFMADRL scheme. This is because CEFMR scheme employs elastic federated learning to train the AAE, which allows individual UE's personalized models to better reflect their own data. This means that for each UE, the personalized AAE model is more likely to provide more accurate predictions of content popularity. When the SBS's cache capacity increases, more accurate predictions can help cache more suitable content, thereby increasing cache hit rates and reducing costs.

Figs. 8-(a), (b) and (c) show the fluctuation figures of the total cost of all SBSs, reward and cache hit ratio with respect to episodes in the testing stage of MADDPG under different cache capacities for each SBS. It can be seen that the performance of the MADDPG tends to be smooth in different testing episodes, which demonstrates the stability of the algorithm. From these figures, it can also be seen that cost decreases as cache capacity increases, while reward and cache hit ratio increase as cache capacity increases. This is because as the cache capacity increases, SBS can cache more contents requested by UEs, and UEs can fetch more contents from the local SBS, thereby improving the cache hit rate. Moreover, as cache capacity increases, UEs can fetch more contents from the local SBSs or from adjacent SBSs, instead of fetching them from CS. Therefore, increasing cache capacity can reduce cost and increase the saved cost, i.e., the reward.

Figs 9-(a), (b), and (c) show the variations in cost, reward, and cache hit ratio with respect to the cache capacity of each SBS under different numbers of SBSs. When the number of SBSs is 1, this case provides the worst performance. This is because the CEFMR scheme has become a single-agent scheme. The local SBS cannot obtain requested contents from

the adjacent SBSs, thus the UEs can only fetch the requested contents from the local SBS or CS with a higher cost. As the number of SBSs increases, the CEFMR scheme generally performs better in terms of cost, reward, and cache hit ratio. This is because more SBSs can cooperate to cache popular contents. It also indicates the effectiveness of the CEFMR scheme in coordinating edge caching.

VI. Conclusions

In this paper, we proposed a cooperative edge caching scheme named CEFMR to optimize the cost to fetch contents in next-generation networks. We first proposed an elastic FL algorithm to train the personalized model for each UE, where the AAE model was adopted for training to improve the prediction accuracy, then a popular content prediction algorithm was proposed to predict the popular contents for each SBS based on the trained AAE model. Finally, we proposed a MADRL based algorithm to decide where the predicted popular contents are collaboratively cached among SBSs to reduce the cost for fetching contents. Our experimental results demonstrated the superiority of our proposed scheme to existing baseline caching schemes. The conclusions are summarized as follows:

- CEFMR's ability to extract hidden features from the local data of UEs significantly outperforms baseline schemes in predicting popular content for caching. Unlike other schemes, CEFMR's personalized approach personalizes predictions to the specific usage patterns and preferences of each UE. This leads to a more accurate and efficient caching process, as evidenced by a comparison without AAE model, which shows the improvements in the cache hit ratio.
- The elastic FL algorithm within CEFMR uniquely assigns specific weights to the model of each UE based on the distance between previously trained local models and the global model. This personalization ensures that each UE's model accurately reflects its data characteristics, a feature not commonly seen in other schemes like traditional FL. This approach has demonstrated CEFMR can enhance the efficiency of the cached content and save cost.
- The MADDPG algorithm in CEFMR facilitates cooperative decision-making on whether to cache content in local SBS, adjacent SBSs or CS. This not only optimizes cache storage distribution but also significantly reduces operational costs compared to the schemes without MADRL. Our comparative analysis reveals that this method leads to improvements in cache efficiency and cost savings.

References

[1] X. Wang, Y . Han, C. Wang, Q. Zhao, X. Chen and M. Chen, "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Netw*, vol. 33, no. 5, pp. 156–165, Sept.-Oct. 2019, doi: 10.1109/MNET.2019.1800286.

[2] Q. Wu, Y. Zhao and Q. Fan, "Time-Dependent Performance Modeling for Platooning Communications at Intersection," *IEEE Internet Things*, vol. 9, no. 19, pp. 18500-18513, 1 Oct.1, 2022, doi: 10.1109/JIOT.2022.3161028.

[3] T. Li, P. Fan, Z. Chen and K. B. Letaief, "Optimum Transmission Policies for Energy Harvesting Sensor Networks Powered by a Mobile Control Center," in *IEEE Transactions on Wireless Communications*, vol. 15, no. 9, pp. 6132-6145, Sept. 2016, doi: 10.1109/TWC.2016.2578925.

[4] Z. Yao, J. Jiang, P. Fan, Z. Cao and V. O. K. Li, "A neighbor-table-based multipath routing in ad hoc networks," *The 57th IEEE Semiannual Vehicular Technology Conference*, 2003. VTC 2003-Spring., Jeju, Korea (South), 2003, pp. 1739-1743 vol.3, doi: 10.1109/VETECS.2003.1207121.

[5] Q. Wu, S. Xia, Q. Fan and Z. Li, "Performance Analysis of IEEE 802.11p for Continuous Backoff Freezing in IoV, " *Electronics*, Vol. 8, No. 1404, Dec. 2019.

[6] J. Zhang, P. Fan and K. B. Letaief, "Network Coding for Efficient Multicast Routing in Wireless Ad-hoc Networks," in *IEEE Transactions on Communications*, vol. 56, no. 4, pp. 598-607, April 2008, doi: 10.1109/TCOMM.2008.060238.

[7] E. Bastug, M. Bennis and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Commun. Mag*, vol. 52, no. 8, pp. 82-89, Aug. 2014, doi: 10.1109/MCOM.2014.6871674.

[8] A. Tian, B. Feng, H. Zhou, Y. Huang, K. Sood, S. Yu and H. Zhang, "Efficient Federated DRL-Based Cooperative Caching for Mobile Edge Networks," *IEEE Transactions on Network and Service Management*, vol. 20, no. 1, pp. 246-260, March 2023, doi: 10.1109/TNSM.2022.3198074.

[9] Y. Zhang, Y. Zhou, S. Zhang, G. Gui, B. Adebisi, H. Gacanin and H. Sari. An Efficient Caching and Offloading Resource Allocation Strategy in Vehicular Social Networks[J]. IEEE Transactions on Vehicular Technology, 2024, 73(4): 5690-5703.

[10] J. Li, H. Chen, Y. Chen, Z. Lin, B. Vucetic and L. Hanzo, "Pricing and Resource Allocation via Game Theory for a Small-Cell Video Caching System," *IEEE J. Sel. Areas Commun*, vol. 34, no. 8, pp. 2115-2129, Aug. 2016, doi: 10.1109/JSAC.2016.2577278.

[11] G. Paschos, E. Bastug, I. Land, G. Caire and M. Debbah, "Wireless caching: technical misconceptions and business barriers," *IEEE Commun. Mag*, vol. 54, no. 8, pp. 16-22, August 2016, doi: 10.1109/MCOM.2016.7537172.

[12] M. Yan, C. A. Chan, W. Li, L. Lei, A. F. Gygax and C. -L. I, "Assessing the Energy Consumption of Proactive Mobile Edge Caching in Wireless Networks," *IEEE Access*, vol. 7, pp. 104394-104404, 2019, doi: 10.1109/ACCESS.2019.2931449.

[13] Y. Chen, Y. Ning, M. Slawski and H. Rangwala, "Asynchronous Online Federated Learning for Edge Devices with Non-IID Data," *2020 IEEE International Conference on Big Data (Big Data)*, Atlanta, GA, USA, 2020, pp. 15-24, doi: 10.1109/BigData50022.2020.9378161.

[14] A. Hammoud, H. Otrok, A. Mourad and Z. Dziong, "On Demand Fog Federations for Horizontal Federated Learning in IoV," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 3062-3075, Sept. 2022, doi: 10.1109/TNSM.2022.3172370.

[15] Y. Gao, L. Liu, X. Zheng, C. Zhang and H. Ma, "Federated Sensing: Edge-Cloud Elastic Collaborative Learning for Intelligent Sensing," *IEEE IoT-J*, vol. 8, no. 14, pp. 11100-11111, July. 2021, doi: 10.1109/JIOT.2021.3053055.

[16] NR and NG-RAN Overall Description, *document 3GPP Technical Specification TS 38.300*, Release 15, 2019.

[17] M. F. Pervej, L. T. Tan and R. Q. Hu, "User Preference Learning-Aided Collaborative Edge Caching for Small Cell Networks," *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, Taipei, Taiwan, 2020, pp. 1-6, doi: 10.1109/GLOBECOM42002.2020.9322208.

[18] L. Yao, Y. Wang, Q. Xia and R. Xu, "Popularity Prediction Caching Using Hidden Markov Model for Vehicular Content Centric Networks," 2019 20th IEEE International Conference on Mobile Data Management (MDM), Hong Kong, China, 2019, pp. 533-538, doi: 10.1109/MDM.2019.00115.

[19] Y. Tao, Y. Jiang, F. -C. Zheng, M. Bennis and X. You, "Content Popularity Prediction in Fog-RANs: A Bayesian Learning Approach," *2021 IEEE Global Communications Conference (GLOBECOM)*, Madrid, Spain, 2021, pp. 1-6, doi: 10.1109/GLOBECOM46510.2021.9685947.

[20] Y. Liu, T. Zhi, H. Xi, X. Duan and H. Zhang, "A Novel Content Popularity Prediction Algorithm Based on Auto Regressive Model in Information-Centric IoT," in *IEEE Access*, vol. 7, pp. 27555-27564, 2019, doi: 10.1109/ACCESS.2019.2901525.

[21] M. Feng, M. Yu and R. Li, "Multi-feature content popularity prediction algorithm based on GRU-Attention in V-NDN," *2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, Rio de Janeiro, Brazil, 2023, pp. 1136-1141, doi: 10.1109/CSCWD57460.2023.10152582.

[22] M. F. Pervej, L. T. Tan and R. Q. Hu, "Artificial Intelligence Assisted Collaborative Edge Caching in Small Cell Networks," *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, Taipei, Taiwan, 2020, pp. 1-7, doi: 10.1109/GLOBECOM42002.2020.9322101.

[23] M. Saputra, D. T. Hoang, D. N. Nguyen and E. Dutkiewicz, "JOCAR: A Jointly Optimal Caching and Routing Framework for Cooperative Edge Caching Networks," *2019 IEEE Global Communications Conference (GLOBECOM)*, Waikoloa, HI, USA, 2019, pp. 1-6, doi: 10.1109/GLOBECOM38437.2019.9013745.

[24] H. Xie, J. Xiong, L. Gui, B. Li and J. Li, "On User Cooperative Caching by Reverse Auction," *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, Honolulu, HI, USA, 2019, pp. 1-5, doi: 10.1109/VTCFall.2019.8891441.

[25] C. Li, S. Gong, Z. Ning, L. Zhao, Q. Jiang and P. Zhou, "Robust Caching Control in Crowdsourced Content-Centric Mobile Networking," in *IEEE Access*, vol. 6, pp. 59811-59821, 2018, doi: 10.1109/ACCESS.2018.2875060.

[26] I. Liao, K. -K. Wong, Y. Zhang, Z. Zheng and K. Yang, "MDS Coded Cooperative Caching for Heterogeneous Small Cell Networks," *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Singapore, 2017, pp. 1-7, doi: 10.1109/GLOCOM.2017.8254854.

[27] C. Fang, C. Liu, Z. Wang, Y. Sun, W. Ni, P. Li and S. Guo, "Cache-Assisted Content Delivery in Wireless Networks: A New Game Theoretic Model," in *IEEE Systems Journal*, vol. 15, no. 2, pp. 2653-2664, June 2021, doi: 10.1109/JSYST.2020.3001229.

[28] S. Araf, A. S. Saha, S. H. Kazi, N. H. Tran and M. G. R. Alam, "UAV Assisted Cooperative Caching on Network Edge Using Multi-Agent Actor-Critic Reinforcement Learning," *IEEE Trans. Veh. Technol*, vol. 72, no. 2, pp. 2322-2337, Feb. 2023, doi: 10.1109/TVT.2022.3209079.

[29] S. Chen, Z. Yao, X. Jiang, J. Yang and L. Hanzo, "Multi-Agent Deep Reinforcement Learning-Based Cooperative Edge Caching for Ultra-Dense Next-Generation Networks," *IEEE Trans. Commun*, vol. 69, no. 4, pp. 2441-2456, April 2021, doi: 10.1109/TCOMM.2020.3044298.

[30] W. Jiang, G. Feng, S. Qin, T. S. P. Yum and G. Cao, "Multi-Agent Reinforcement Learning for Efficient Content Caching in Mobile D2D Networks," *IEEE Trans. Wireless Commun*, vol. 18, no. 3, pp. 1610-1622, March 2019, doi: 10.1109/TWC.2019.2894403.

[31] X. Wu, J. Li, M. Xiao, P. C. Ching and H. V. Poor, "Multi-Agent Reinforcement Learning for Cooperative Coded Caching via Homotopy Optimization," *IEEE Trans. Wireless Commun*, vol. 20, no. 8, pp. 5258-5272, Aug. 2021, doi: 10.1109/TWC.2021.3066458.

[32] C. Zhong, M. C. Gursoy and S. Velipasalar, "Deep Reinforcement Learning-Based Edge Caching in Wireless Networks," *IEEE Trans. Cogn. Commun. Netw*, vol. 6, no. 1, pp. 48-61, March 2020, doi: 10.1109/TCCN.2020.2968326.

[33] K. Jiang, H. Zhou, D. Zeng and J. Wu, "Multi-Agent Reinforcement Learning for Cooperative Edge Caching in Internet of Vehicles," *2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, Delhi, India, 2020, pp. 455-463, doi: 10.1109/MASS50613.2020.00062.

[34] Q. Wu, S. Wang, H. Ge, P. Fan, Q. Fan and K. B. Letaief, "Delay-Sensitive Task Offloading in Vehicular Fog Computing-Assisted Platoons," in IEEE Transactions on Network and Service Management, vol. 21, no. 2, pp. 2012-2026, April 2024, doi: 10.1109/TNSM.2023.3322881.

[35] Q. Wu, W. Wang, P. Fan, Q. Fan, J. Wang and K. B. Letaief, "URLLC-Awared Resource Allocation for Heterogeneous Vehicular Edge Computing,"*IEEE Trans. Veh. Technol.*, 2024, doi: 10.1109/TVT.2024.3370196.

[36] Q. Wu, S. Shi, Z. Wan, Q. Fan, P. Fan and C. Zhang, "Towards V2I Age-aware Fairness Access: A DQN Based Intelligent Vehicular Node Training and Test Method", Chinese Journal of Electronics, vol. 32, no. 6, 2023, pp. 1230-1244.

[37] J. Song, M. Sheng, T. Q. S. Quek, C. Xu and X. Wang, "Learning-Based Content Caching and Sharing for Wireless Networks," *IEEE Trans. Commun*, vol. 65, no. 10, pp. 4309-4324, Oct. 2017, doi: 10.1109/TCOMM.2017.2713384.

[38] X. Cai, J. Zheng, Y. Fu, Y. Zhang and W. Wu, "Cooperative content caching and delivery in vehicular networks: A deep neural network approach," *China Commun*, vol. 20, no. 3, pp. 43-54, March 2023, doi: 10.23919/JCC.2023.03.004.

[39] F. Wang, F. Wang, J. Liu, R. Shea and L. Sun, "Intelligent Video Caching at Network Edge: A Multi-Agent Deep Reinforcement Learning Approach," *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, Toronto, ON, Canada, 2020, pp. 2499-2508, doi: 10.1109/INFOCOM41043.2020.9155373.

[40] Y. Chen, Y. Cai, H. Zheng, J. Hu and J. Li, "Cooperative caching for scalable video coding using value-decomposed dimensional networks," *China Commun*, vol. 19, no. 9, pp. 146-161, Sept. 2022, doi: 10.23919/JCC.2022.00.006.

[41] D. Zhang, W. Wang, J. Zhang, T. Zhang, J. Du and C. Yang, "Novel Edge Caching Approach Based on Multi-Agent Deep Reinforcement Learning for Internet of Vehicles," *IEEE Trans. Intell. Transp. Syst*, doi: 10.1109/TITS.2023.3264553.

[42] J. Chang, N. Zhang, M. Tao and H. Tuo, "Federated Multi-Agent Reinforcement Learning for Collaborative Edge Caching in Content Delivery Networks," *2022 14th International Conference on Wireless Communications and Signal Processing (WCSP)*, Nanjing, China, 2022, pp. 166-170, doi: 10.1109/WCSP55476.2022.10039249.

[43] S. B. Prathiba, G. Raja, S. Anbalagan, S. Gurumoorthy, N. Kumar and M. Guizani, "Cybertwin-Driven Federated Learning Based Personalized Service Provision for 6G-V2X," *IEEE Trans. Veh. Technol*, vol. 71, no. 5, pp. 4632-4641, May 2022, doi: 10.1109/TVT.2021.3133291.

[44] D. Sun, X. Li, J. Wen, X. Wang, Z. Han and V. C. M. Leung, "Federated Deep Reinforcement Learning for Recommendation-Enabled Edge Caching in Mobile Edge-Cloud Computing Networks," in IEEE Journal on Selected Areas in Communications, vol. 41, no. 3, pp. 690-705, March 2023, doi: 10.1109/JSAC.2023.3235443.

[45] Y. Liu and B. Mao, "On a Novel Content Edge Caching Approach based on Multi-Agent Federated Reinforcement Learning in Internet of Vehicles," 2023 32nd Wireless and Optical Communications Conference (WOCC), Newark, NJ, USA, 2023, pp. 1-5, doi: 10.1109/WOCC58016.2023.10139417.

[46] W. Liu, H. Zhang, H. Ding, D. Li and D. Yuan, "Mobility-Aware Coded Edge Caching in Vehicular Networks with Dynamic Content Popularity," *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, Nanjing, China, 2021, pp. 1-6, doi: 10.1109/WCNC49053.2021.9417383.

[47] Z. Yu, J. Hu, G. Min, Z. Zhao, W. Miao and M. S. Hossain, "Mobility-Aware Proactive Edge Caching for Connected Vehicles Using Federated Learning," *IEEE Trans. Intell. Transp. Syst*, vol. 22, no. 8, pp. 5341-5351, Aug. 2021, doi: 10.1109/TITS.2020.3017474.

[48] Q. Wu, X. Wang, Q. Fan, P. Fan, C. Zhang and Z. Li, "High Stable and Accurate Vehicle Selection Scheme based on Federated Edge Learning in Vehicular Networks", China Communications, Vol. 20, No. 3, Mar. 2023, pp. 1-17.

[49] D. Long, Q. Wu, Q. Fan, P. Fan, Z. Li and J. Fan, "A Power Allocation Scheme for MIMO-NOMA and D2D Vehicular Edge Computing Based on Decentralized DRL", Sensors, Vol. 23, No. 7, 2023, Art. no. 3449.

[50] S. Manzoor, A. Mian, A. Zoha and M. Imran, et al. "Federated learning empowered mobility-aware proactive content offloading framework for fog radio access networks." Future Generation Computer Systems 133 (2022): 307-319.

[51] Q. Wu, Y. Zhao, Q. Fan, P. Fan, J. Wang and C. Zhang, "Mobility-Aware Cooperative Caching in Vehicular Edge Computing Based on Asynchronous Federated and Deep Reinforcement Learning," *IEEE J. Sel. Topics Signal Process*, vol. 17, no. 1, pp. 66-81, Jan. 2023, doi: 10.1109/JSTSP.2022.3221271.

[52] M. Parvini, M. R. Javan, N. Mokari, B. Abbasi and E. A. Jorswieck, "AoI-Aware Resource Allocation for Platoon-Based C-V2X Networks via Multi-Agent Multi-Task Reinforcement Learning," *IEEE Trans. Veh. Technol*, doi: 10.1109/TVT.2023.3259688.

[53] H. Zhu, Q. Wu, X.-J. Wu, Q. Fan, P. Fan, and J. Wang, "Decentralized power allocation for MIMO-NOMA vehicular edge computing based on deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 12 770–12 782, 2022, doi: 10.1109/JIOT.2021.3138434.

[54] T. Nie, J. Luo, L. Gao, F. -C. Zheng and L. Yu, "Cooperative Edge Caching in Small Cell Networks with Heterogeneous Channel Qualities," *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, Antwerp, Belgium, 2020, pp. 1-6, doi: 10.1109/VTC2020-Spring48590.2020.9128365.

[55] W. Wang, Y. Zhao, Q. Wu, Q. Fan, C. Zhang and Z. Li, "Asynchronous Federated Learning Based Mobility-aware Caching in Vehicular Edge Computing," *2022 14th International Conference on Wireless Communications and Signal Processing (WCSP)*, Nanjing, China, 2022, pp. 1-5, doi: 10.1109/WCSP55476.2022.10039430.

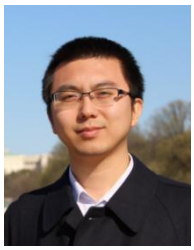
Qiong Wu (Senior Member, IEEE) received the Ph.D. degree in information and communication engineering from National Mobile Communications Research Laboratory, Southeast University, Nanjing, China, in 2016. From 2018 to 2020, he was a postdoctoral researcher with the Department of Electronic Engineering, Tsinghua University, Beijing, China. He is currently an associate professor with the School of Internet of Things Engineering, Jiangnan University, Wuxi, China.

Dr. Wu is a Senior Member of IEEE. He has published over 60 papers in high impact journals and conferences and authorized over 20 patents. He has received the young scientist award for ICCCS'24. He has severed as the section board member of Sensors, the early career editorial board member of CMC-Computers Materials & Continua, Radio Engineering, and Chinese Journal on Internet of Things, the lead guest editor of Sensors, CMC-Computers Materials & Continua and Frontiers in Space Technologies, the guest editor of Electronics, the TPC co-chair of WCSP'22, the workshop chair of NCIC'23 and ICFEICT'24, as well as the TPC member and session chair for over 10 international Conferences. His current research interest focuses on vehicular networks, autonomous driving communication technology, and machine learning.

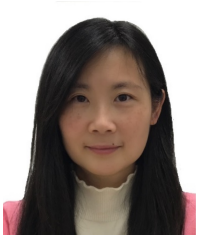
Wenhua Wang received the B.S. degree from the Jiangnan University, Wuxi, China, in 2021. He is currently working toward the M.S. degree with Jiangnan University. His research interests include federated learning, reinforcement learning, and mobile edge computing.

Pingyi Fan (Senior Member, IEEE) received the B.S. degree from the Department of Mathematics, Hebei University, in 1985, the M.S. degree from the Department of Mathematics, Nankai University, in 1990, and the Ph.D. degree from the Department of Electronic Engineering, Tsinghua University, Beijing, China, in 1994. From August 1997 to March 1998, he visited The Hong Kong University of Science and Technology as a Research Associate. From May 1998 to October 1999, he visited the University of Delaware, Newark, DE, USA, as a Research Fellow. In March. 2005, he visited NICT, Japan, as a Visiting Professor. From June 2005 to May 2014, he visited The Hong Kong University of Science and Technology for many times. From July 2011 to September 2011, he was a Visiting Professor at the Institute of Network Coding, The Chinese University of Hong Kong. He is currently a Professor and the director of open source data recognition innovation center at the Department of Electrical Engineering (EE), Tsinghua University. His main research interests include B5G technology in wireless communications, such as MIMO, OFDMA, network coding, network information theory, machine learning, and big data analysis.

Dr. Fan is a Fellow of IET and an Overseas Member of IEICE. He is also a reviewer of more than 40 international journals, including 30 IEEE journals and eight EURASIP journals. He has received some academic awards, including the IEEE WCNC'08 Best Paper Award, the IEEE ComSoc Excellent Editor Award for IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS in 2009, the ACM IWCMC'10 Best Paper Award, the IEEE Globecom'14 Best Paper Award, the IEEE ICC'20 Best Paper Award, the IEEE TAOS Technical Committee'20 Best Paper Award, IEEE ICCCS Best Paper Awards in 2023 and 2024, the CIEIT Best Paper Awards in 2018 and 2019. He has served as an editor of IEEE Transactions on Wireless Communications, Inderscience International Journal of Ad Hoc and Ubiquitous Computing and Wiley Journal of Wireless Communication and Mobile Computing, MDPI Electronics and Open Journal of Mathematical Sciences etc. He has attended to organize many international conferences, as the general chairs/TPC chair/ Plenary/Keynote speaker for over 30 international Conferences, including as the General Co-Chair of EAI Chinacom2020, 2023 and IEEE VTS HMWC 2014, the TPC Co-Chair of IEEE ICCCS2024 and a TPC Member of IEEE ICC, Globecom, WCNC, VTC, and Inforcom. He has served as an Editor for IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, International Journal of Ad Hoc and Ubiquitous Computing (Inderscience), Journal of Wireless Communication and Mobile Computing (Wiley), Electronics (MDPI), and Open Journal of Mathematical Sciences, IAES international journal of artificial intelligence. He is also an associate editor of IEEE Transactions on cognitive communications and networking.



Qiang Fan received his Ph.D. degree in Electrical and Computer Engineering from New Jersey Institute of Technology (NJIT) in 2019, and his M.S. degree in Electrical Engineering from Yunnan University of Nationalities, China, in 2013. He was a postdoctor researcher in the Department of Electrical and Computer Engineering, Virginia Tech. Currently, he is a staff engineer in Qualcomm, USA. He has served as a reviewer for over 120 journal submissions such as IEEE Transactions on Cloud Computing, IEEE Journal on Selected Areas in Communications, IEEE Transactions on Communications. His current research interests include wireless communications and networking, mobile edge computing, machine learning and drone assisted networking.



JHuling Zhu (SM'17) received the B.S degree from Xidian University, China, and the Ph.D. degree from Tsinghua University, China. She is currently a Reader (Associate Professor) in the School of Engineering, University of Kent, United Kingdom. Her research interests are in the area of wireless communications. She was holding European Commission Marie Curie Fellowship from 2014 to 2016. She received the best paper award from IEEE Globecom 2011. She was Symposium Co-Chair for IEEE Globecom 2015 and IEEE ICC 2018, and Track Co-Chair of IEEE VTC2016-Spring and VTC2018-Spring. Currently, she serves as an Editor for IEEE Transactions on Vehicular Technology.

Khaled Ben Letaief (Fellow, IEEE) received the B.S. (Hons.), M.S., and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, USA, in December 1984, August 1986, and May 1990, respectively.

From 1990 to 1993, he was a Faculty Member with The University of Melbourne, Melbourne, VIC, Australia. Since 1993, he has been with The Hong Kong University of Science and Technology (HKUST). While at HKUST, he has held many administrative positions, including the Head of the Electronic and Computer Engineering Department, the Director of the Wireless IC Design Center, the Founding Director of Huawei Innovation Laboratory, and the Director of Hong Kong Telecom Institute of Information Technology. While at HKUST, he has also served as a Chair Professor and the Dean of Engineering. Under his leadership, the School of Engineering has not only transformed its education and scope and produced very high caliber scholarship, but it has also actively pursued knowledge transfer and societal engagement in broad contexts. It has also dazzled in international rankings (rising from #26 in 2009 to #14 in the world in 2015 according to QS World University Rankings). From September 2015 to March 2018, he joined HBKU as a Provost to help establish a research-intensive university in Qatar in partnership with strategic partners that include Northwestern University, Carnegie Mellon University, Cornell, and Texas A&M. He served as Consultants for different organizations, including Huawei, ASTRI, ZTE, Nortel, PricewaterhouseCoopers, and Motorola. He is currently with the Peng Cheng Laboratory, Shenzhen, China. He is also an Internationally Recognized Leader in wireless communications and networks with research interests in artificial intelligence, big data analytics systems, mobile cloud, and edge computing, tactile internet, and 5G systems. In these areas, he has over 630 articles with over 38,350 citations and an H-index of 87 along with 15 patents, including 11 U.S. inventions.

Dr. Letaief is a member of the United States National Academy of Engineering, a fellow of Hong Kong Institution of Engineers, and a member of Hong Kong Academy of Engineering Sciences. He was a recipient of many distinguished awards and honors, including the 2019 Distinguished Research Excellence Award by HKUST School of Engineering (Highest Research Award and only one recipient/three years is honored for his/her contributions), the 2019 IEEE Communications Society and Information Theory Society Joint Paper Award, the 2018 IEEE Signal Processing Society Young Author Best Paper Award, the 2017 IEEE Cognitive Networks Technical Committee Publication Award, the 2016 IEEE Signal Processing Society Young Author Best Paper Award, the 2016 IEEE Marconi Prize Paper Award in Wireless Communications, the 2011 IEEE Wireless Communications Technical Committee Recognition Award, the 2011 IEEE Communications Society Harold Sobol Award, the 2010 Purdue University Outstanding Electrical and Computer Engineer Award, the 2009 IEEE Marconi Prize Award in Wireless Communications, the 2007 IEEE Communications Society Joseph LoCicero Publications Exemplary Award, and more than 16 IEEE best paper awards. He is well recognized for his dedicated service to professional societies and IEEE, where he has served in many leadership positions, including the Treasurer of the IEEE Communications Society, the IEEE Communications Society VicePresident for Conferences, the Chair of the IEEE Committee on Wireless Communications, an Elected Member of the IEEE Product Services and Publications Board, and the IEEE Communications Society Vice-President for Technical Activities. He is the Founding Editor-in-Chief of the prestigious IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and has served on the Editorial Board of other premier journals, including the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS: Wireless Communications Series (as the Editor-in-Chief). He has also been involved in organizing many flagship international conferences. He also served as the President of the IEEE Communications Society from 2018 to 2019, the world's leading organization for communications professionals with headquarter in New York City and members in 162 countries. He is also recognized by Thomson Reuters as an ISI Highly Cited Researcher and was listed among the 2020 top 30 of AI 2000 Internet of Things Most Influential Scholars. IEEE Committee on Wireless Communications, an Elected Member of the IEEE Product Services and Publications Board, and the IEEE Communications Society Vice-President for Technical Activities. He is the Founding Editor-in-Chief of the prestigious IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and has served on the Editorial Board of other premier journals, including the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS: Wireless Communications Series (as the Editor-in-Chief). He has also been involved in organizing many flagship international conferences. He also served as the President of the IEEE Communications Society from 2018 to 2019, the world's leading organization for communications professionals with headquarter in New York City and

members in 162 countries. He is also recognized by Thomson Reuters as an ISI Highly Cited Researcher and was listed among the 2020 top 30 of AI 2000 Internet of Things Most Influential Scholars.