A Novel Deep Federated Learning-Based and Profit-Driven Service Caching Method

Zhaobin Ouyang[1], Yunni Xia[1(✉)], Qinglan Peng[2], Yin Li[3], Peng Chen[4(✉)], and Xu Wang[5]

[1] College of Computer Science, Chongqing University, Chongqing, China
xiayunni@hotmail.com
[2] School of Artificial Intelligence, Henan University, Zhengzhou, China
[3] Guangzhou Institute of Software Application Technology, Guangzhou, China
[4] School of Computer and Software Engineering, Xihua University, Chengdu, China
chenpeng@mail.xhu.edu.cn
[5] College of Mechanical and Vehicle Engineering, Chongqing University, Chongqing, China

Abstract. Service caching is an emerging solution to addressing massive service request in a distributed environment for supporting rapidly growing services and applications. With the explosive increases in global mobile data traffic, service caching over the edge computing architecture, Mobile edge computing (MEC), emerges for alleviating traffic congestion as well as for optimizing the efficiency of task processing. In this manuscript, we propose a novel profit-driven service caching method based on a federated learning model for service prediction and a deep reinforcement learning mode for yielding caching decisions (FPDRD) in an edge environment. The proposed method is temporal service popularity and user preference-aware. It aims to ensure quality of service (QoS) of delivery of cached service while maximizing the profits of network service providers. Experimental results clearly demonstrate that the FPDRD method outperforms traditional methods in multiple aspects.

Keywords: service caching · profit maximization · popularity prediction · caching decisions · collaborative mechanism

1 Introduction

In recent years, the explosive growth of mobile applications and the growing need for low-latency and high-bandwidth services have placed significant strain on the traditional cloud-centric network infrastructure [1,2]. To tackle these challenges, edge computing has emerged as a promising paradigm that brings computing and storage capabilities closer to end-users. This proximity allows for Lower latency, minimized network congestion and enhanced quality of service (QoS) [3–5].

Edge service caching is a critical component of edge computing, which significantly contributes to the improvement of mobile application performance 3.

H. Gao et al. (Eds.): CollaborateCom 2023, LNICST 563, pp. 118–133, 2024.
https://doi.org/10.1007/978-3-031-54531-3_7

It involves the strategic storage of frequently accessed data and services on edge servers, referred to as Fog Access Points (FAPs). The purpose of this caching strategy is to reduce delay and alleviate the workload on central cloud data centers. By employing edge service caching, faster access to content and services is made possible, especially for latency-sensitive applications like augmented reality, video streaming and real-time data processing.

However, various challenges in this direction are yet to be properly addressed. Firstly, FAPs are with limited computational and storage resources, thus guaranteeing only a small amount of services is cachable and making hit rate low. Secondly, in a highly dynamic and volatile edge environment, static caching strategies are often inadequate in meeting the changing needs. Nevertheless, how to yield run-time caching decisions according to time-varying service popularity and user needs remains a difficulty. Finally, existing methods in this direction usually aim to optimize caching performance, in terms of hit rate and delivery latency. How to guarantee profit of service providers is less studied.

In this paper, we propose a novel caching method by leveraging a federated learning model for popularity prediction and a deep reinforcement learning model for yielding caching decisions (FPDRD). The FPDRD method takes both global service popularity and local user preferences as inputs and can achieve reasonable tradeoffs between caching performance and profit of providers. Extensive simulations are conducted based on a well-known dataset, Movielens. Numerical results clearly indicate that our proposed method outperform it peers.

The paper is organized as follows: Section 2 provides a literature review. Section 3 presents the system model and the problem formulation. Section 4 describes the proposed method. Section 5 presents the empirical analysis.

2 Related Work

Task offloading and caching in MEC have gained significant attention in recent years as a means to alleviate the resource constraints faced by FAPs. In their work, Gao *et al.* [7] presented a method that combines task offloading scheduling and resource allocation to minimize task delay and energy consumption. Liu *et al.* [8] proposed an approach that utilizes online computation offloading and resource scheduling to tackle the challenges arising from user mobility and network dynamics.

Due to resource and energy constraints, FAPs are usually allowed to cache limited services [9]. Thus, caching of highly popular services in FAPs has emerged as an effective solution when FAPs are limited in caching capacity. Zhong *et al.* [10] proposed the Cocktail Edge Caching method, which utilizes an ensemble learning algorithm to predict the popularity of services. However, this approach only takes into account the overall service popularity while neglecting the preferences of local users. In contrast, Li *et al.* [11] propose a service caching method that considers hit actions and user perception preferences. However, this approach raises concerns regarding user privacy and security, as it shares all users' personal information for prediction purposes.

Recently, the cooperative service caching mechanism was proposed as well for exploiting multiple cache nodes through making them working together [12–15]. The problem of cooperative service caching can be formulated as a Mixed-Integer Nonlinear Programming (MINLP) problem, which is known to be inherently NP-hard [16,17]. Wu *et al.* [18] and Xu *et al.* [19] propose a coalition formation algorithm that utilizes a hedonic game among cooperative service providers for optimizing both the overall profit of the coalition and the average profit of each individual participant. Li *et al.* [20] propose a DRL algorithm-based profit-driven cooperative service placement method in MEC.

3 System Models and Problem Formulation

Here, we consider a service caching system in MEC represented as $\mathcal{G} = (\mathcal{CCS} \cup \mathcal{N} \cup \mathcal{U} \cup \mathcal{L})$, as illustrated in Fig. 1. This system consists of a central cloud server (CCS), multiple fog access points (FAPs) denoted as $\mathcal{N} = \{\mathcal{N}_1, \mathcal{N}_2, ..., \mathcal{N}_n\}$, multiple users denoted as $\mathcal{U} = \{\mathcal{U}_1, \mathcal{U}_2, ..., \mathcal{U}_u\}$ and a set of service types denoted as $\mathcal{L} = \{\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_l\}$. Network edge service providers operate each FAP, which is equipped with storage, computing and communication capabilities. Users are provided services by these FAPs by using a billing mechanism.

3.1 Caching Model

Due to capacity and storage limits, FAPs cache a subset of application services. These cached applications require periodic updates and replacement. The caching decisions is represented by a binary variable $x_{n,l}$, which can be expressed as:

$$x_{n,l} = \begin{cases} 1, & \text{if service } \mathcal{L}_l \text{ is cached in the FAP } \mathcal{N}_n, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The constraint on the storage space at the FAP $\mathcal{N}_n$ is:

$$\sum_{l \in \mathcal{L}} x_{n,l} \omega_l \leq \Omega_n \tag{2}$$

where ω_l represents the storage capacity required for service $\mathcal{L}_l$ and Ω_n the total cache capacity of FAP $\mathcal{N}_n$.

Users are charged for use of service $\mathcal{L}_l$. Such charge is proportional to use time:

$$F_l = t_l \cdot p_l \tag{3}$$

where p_l represent the price charged by the service provider for the execution of service $\mathcal{L}_l$ per unit of time.

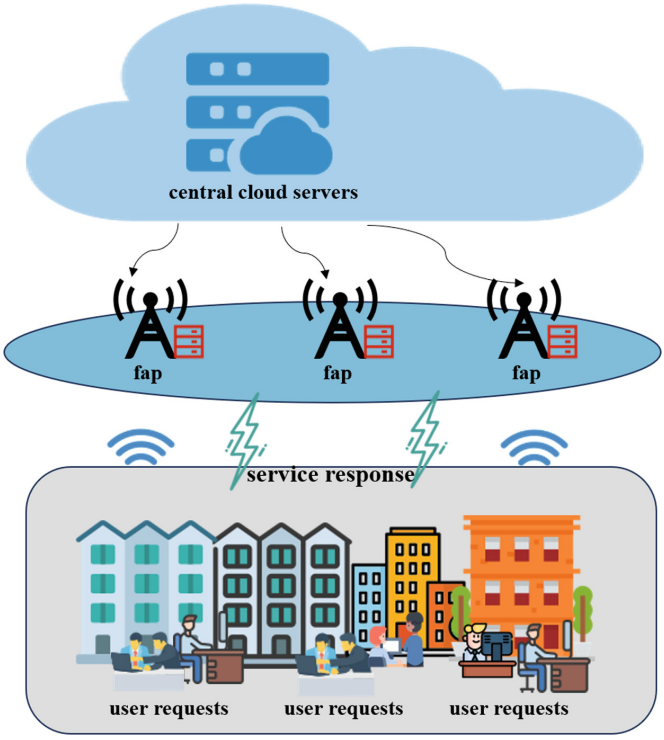


Fig. 1. System model.

3.2 Computation and Communication Model

According to a collaborative service cache mechanism (CSCM) [21], where FAP $\mathcal{N}_n$ receives a service request from a user. Initially, FAP $\mathcal{N}_n$ checks its local cache to determine if service $\mathcal{L}_l$ is cached. If the service is cached locally, FAP $\mathcal{N}_n$ handles the user's service request directly. Otherwise, FAP $\mathcal{N}_n$ seeks cached services from neighboring FAPs. When multiple FAPs cache service $\mathcal{L}_l$, FAP $\mathcal{N}_n$ turns to FAP $\mathcal{N}_m$ with the lowest transmission cost. When none of the FAPs caches the required service $\mathcal{L}_l$, the user's service request is forwarded to the CCS. The cost $C^x_{n,l}$ represents the charge by FAP $\mathcal{N}_n$ when processing service $\mathcal{L}_l$ through server x:

$$C^x_{n,l} = \beta_1 \cdot L^x_{n,l} + \beta_2 \cdot E^x_{n,l} \tag{4}$$

where $L^x_{n,l}$ and $E^x_{n,l}$ denote the delay and energy consumption when FAP $\mathcal{N}_n$ processes service $\mathcal{L}_l$ through server x, respectively. β_1 and β_2 indicate the economic factors associated with the delay and energy consumption, respectively.

When FAP $\mathcal{N}_n$ receives a service request from a user, it first checks whether service $\mathcal{L}_l$ is cached locally. In this case, FAP $\mathcal{N}_n$ directly processes the user's service request. In such a scenario, FAP $\mathcal{N}_n$ places the service onto the thread

of the queue with the shortest waiting time for processing. In this context, the local delay and energy consumption for service $\mathcal{L}_l$ are:

$$L_{n,l}^{local} = (\sum_{o \in q_n} \zeta_o + \zeta_l) \cdot t_c + L_{base} \tag{5a}$$

$$E_{n,l}^{local} = \zeta_l f_n^2 \epsilon_n + E_{base} \tag{5b}$$

where q_n represents the set of tasks in the queue with the shortest waiting time on FAP $\mathcal{N}_n$, ζ_l the computational workload, t_c the unit processing time for a task and f_n the computing capacity of FAP $\mathcal{N}_n$.

In case that service is not cached locally at FAP $\mathcal{N}_n$, the FAP $\mathcal{N}_n$ turns to other servers. According to Shannon's formula, the transmission rate between them is:

$$r_{i,j} = B_i \log_2 \left(1 + \frac{P_i |h_i|^2}{\sigma^2}\right) \tag{6}$$

where B_i represents the bandwidth rate, P_i the transmission energy consumption, $|h_i|^2$ the channel gain and σ^2 the variance of the additive white Gaussian noise (AWGN).

In case that FAP $\mathcal{N}_n$ finds one or more servers that cache service $\mathcal{L}_l$ among the neighboring FAPs, it chooses the FAP $\mathcal{N}_m$ with the lowest cost for processing the user's service request. In this case, the delay and energy consumption for executing service $\mathcal{L}_l$ are:

$$L_{n,l}^{m} = (\sum_{o \in q_m} \zeta_o + \zeta_l) \cdot t_c + \frac{d_{m,n}^l}{r_{m,n}} + L_{base} \tag{7a}$$

$$E_{n,l}^{m} = \zeta_l f_m^2 \epsilon_m + P_m \frac{d_{m,n}^l}{r_{m,n}} + E_{base} \tag{7b}$$

Where $d_{m,n}^l$ represents the bit size of the computed result of the service request l that is processed by FAP $\mathcal{N}_m$ and forwarded to FAP $\mathcal{N}_n$.

In case that neither FAP $\mathcal{N}_n$ nor its neighboring FAPs cache service $\mathcal{L}_l$, the service request is offloaded to the CCS. In this case, the delay and energy consumption for executing service $\mathcal{L}_l$ are:

$$L_{n,l}^{ccs} = \zeta_l \cdot t_c + \frac{d_{ccs,n}^l}{r_{ccs,n}} + L_{base} \tag{8a}$$

$$E_{n,l}^{ccs} = \zeta_l f_{ccs}^2 \epsilon_{ccs} + P_{ccs} \frac{d_{ccs,n}^l}{r_{ccs,n}} + E_{base} \tag{8b}$$

For each service request, the profit obtained by FAP $\mathcal{N}_n$ can be calculated as the gap between the service request's fee and the total cost incurred. The overall cost is comprised of the cost of caching service $C_{x,l}$ in the server x, the cost of collaboration $C_{n,l}^x$ with other servers x and the equipment-related baseline cost C_{base}. Thus, profit of service request is:

$$V_{n,l}^{local} = F_l - C_{n,l} - C_{n,l}^{\mathcal{N}_n} - C_{base} \tag{9a}$$

$$V_{n,l}^{m} = F_l - C_{m,l} - C_{n,l}^{\mathcal{N}_m} - C_{base} \tag{9b}$$

$$V_{n,l}^{ccs} = F_l - C_{ccs,l} - C_{n,l}^{CCS} - C_{base} \tag{9c}$$

Consequently, the total profit earned by FAP $\mathcal{N}_n$ is:

$$V_n = \sum_{l \in \mathcal{L}} \left(P_{n,l}^{local} \cdot V_{n,l}^{local} + P_{n,l}^{\mathcal{N}_m} \cdot V_{n,l}^{m} + P_{n,l}^{ccs} \cdot V_{n,l}^{ccs} \right) \tag{10}$$

where $P_{n,l}^{local}$, $P_{n,l}^{\mathcal{N}_m}$ and $P_{n,l}^{ccs}$ are binary variables represent the execution modes for FAP $\mathcal{N}_n$ to handle service request $\mathcal{L}_l$ as local execution, execution with assistance from FAP $\mathcal{N}_m$ and execution via the CCS, respectively.

Additionally, the QoS of users $\overline{U_l}$ is decided by the average delay $\overline{L_l}$ and average fee $\overline{F_l}$ of executing service $\mathcal{L}_l$:

$$\overline{U_l} = \eta_l \frac{\overline{L_l} - L_{min}}{L_{max} - L_{min}} + \eta_f \frac{\overline{F_l} - F_{min}}{F_{max} - F_{min}} \tag{11}$$

where η_l and η_f represent the impact factors of delay and price on the QoS of users, respectively. L_{max}, F_{max} and L_{min}, F_{min} represent the maximum and minimum of delay and fee for executing service $\mathcal{L}_l$, respectively.

3.3 Problem Formulation

Based on the system model given above we are interested in maximizing profits of service provider with the constraints of caching capacities. According to (10), the profit of FAP $\mathcal{N}_n$ is decide by the local cache hit rate $P_{n,l}^{local}$ and the collaborative cache hit rate $P_{n,l}^{\mathcal{N}_m}$. The resulting optimization formulation is thus:

$$\mathbf{P}: \quad \max \sum_{t \in T} \sum_{\mathcal{N}_n \in \mathcal{N}} V_n \tag{12}$$

$$\textbf{s.t.} \quad \mathbf{C1}: \overline{U_l} < U_{min}, \forall \mathcal{L}_l \in \mathcal{L} \tag{13a}$$

$$\mathbf{C2}: x_{n,l} \in \{0,1\}, \forall \mathcal{N}_n \in \mathcal{N}, \forall \mathcal{L}_l \in \mathcal{I} \tag{13b}$$

$$\mathbf{C3}: \sum_{l \in \mathcal{L}} x_{n,l} \omega_l \leq \Omega_n, \forall \mathcal{N}_n \in \mathcal{N} \tag{13c}$$

$$\mathbf{C4}: P_{n,l}^{local}, P_{n,l}^{\mathcal{N}_m}, P_{n,l}^{ccs} \in \{0,1\}, \forall \mathcal{N}_n, \mathcal{N}_m \in \mathcal{N}, \forall l \in \mathcal{L} \tag{13d}$$

$$\mathbf{C5}: P_{n,l}^{local} +, P_{n,l}^{\mathcal{N}_m} + P_{n,l}^{ccs} = 1 \tag{13e}$$

$$\mathbf{C6}: V_n \geq 0, \forall \mathcal{N}_n \in \mathcal{N} \tag{13f}$$

$$\mathbf{C7}: Num_{n,l} \leq 1 \tag{13g}$$

Constraint (13a) ensures that the QoS of $\mathcal{L}_l$ is bounded. Constraints (13b) and (13c) indicate the limit of total storage capacity of cached services on FAP

$\mathcal{N}_n$. Constraint (13f) guarantees that the profit for each FAP $\mathcal{N}_n$ must be non-negative. Constraint (13g) indicates that each FAP $\mathcal{N}_n$ can cache service $\mathcal{L}_l$ at most once. The above optimization problem is clearly a Mixed-Integer Nonlinear Programming (MINLP) one, which is also NP-hard.

4 The Proposed Method

In this section, we present a detailed description of the FPDRD method. Firstly, we employ a Federated Learning model for accurate prediction of local popularity by taking the global popularity model and user perception preferences as inputs. We maintain a popularity priority queue $Q_c^{\mathcal{N}_n}$ of FAP $\mathcal{N}_N$ and feed the popularity priority queues Q_c for each FAPs as input of into a deep reinforcement learning model. The learning model yields collaborative service caching decisions according to the optimization objective and constraints.

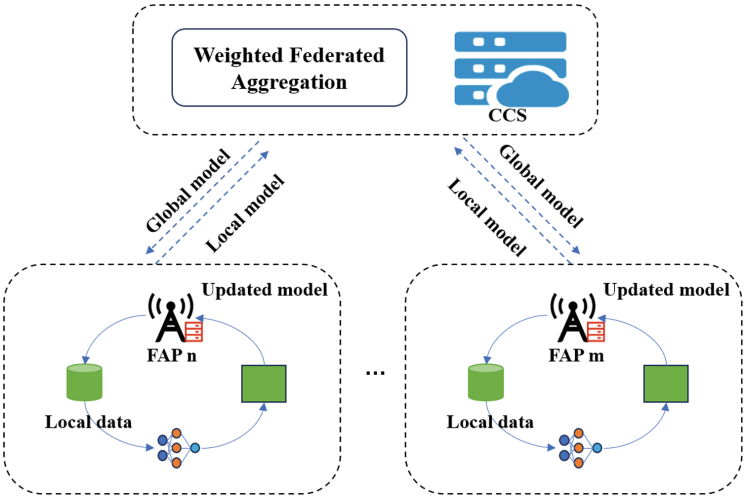


Fig. 2. Popularity prediction model.

4.1 Federated Learning for Popularity Prediction

As shown in Fig. 2, we implement the prediction of popular services based on FL algorithm. The popularity prediction includes the following three steps:

Download Global Model. At the start of each time slot t, every FAP retrieves the global model parameters $\mathbf{W}_t$ from the CCS (Lines 3). These model parameters facilitate the extraction of latent features to predict popular services. This enables each FAP to determine the overall popularity for services during the current time slot.

Local Model Training. Upon receiving the global model parameters $\mathbf{W}_t$, each FAP updates its local model through training iterations (Line 4–8). Subsequently, the updated local model $\mathbf{H}_t^{\mathcal{N}_n}$ is uploaded to the CCS (Line 9). This local model incorporates hidden features related to global service popularity as well as captures hidden features specific to the local users' service perception preferences. By utilizing FAP $\mathcal{N}_n$'s local model $\mathbf{H}_t^{\mathcal{N}_n}$, the local service popularity priority queue $Q_c^{\mathcal{N}_n}$ can be predicted (Line 10). The loss function employed is the categorical cross-entropy, a generalized version of binary cross-entropy, to determine the logarithmic loss for multi-class predictions. This loss function measures the misclassification between the true service $\mathcal{L}_l$ label $\boldsymbol{\pi}$ and the predicted service $\mathcal{L}_l$ label $\hat{\boldsymbol{\pi}}$ defined as cross-entropy:

$$L(\hat{\boldsymbol{\pi}}, \boldsymbol{\pi}) = -\sum_i \boldsymbol{\pi}_i \log(\hat{\boldsymbol{\pi}}_i) \tag{14}$$

Then, we estimate the loss according to its Mean Squared Error (MSE):

$$L(\hat{\phi}, \phi) = \mathbb{E}[(\phi_i - \hat{\phi}_i)^2] \tag{15}$$

where ϕ_i is the real service request label and $\hat{\phi}_i$ the predicted one.

Algorithm 1. Federated Learning for Popularity Prediction

Input: A set of service requests.
Output: The predicted popularity priority queue Q_c.

```
1: for t ∈ 𝒯 do
2:   for 𝒩_n ∈ 𝒩 do
3:     Download the global model W_t.
4:     for the service requests received by 𝒩_n at time t do
5:       Calculate the loss of service ℒ_l according to Eq.(14).
6:     end for
7:     Calculate the loss of FAP 𝒩_n according to Eq.(15).
8:     Update model parameters H_t^{𝒩_n}.
9:     Upload H_t^{𝒩_n} to the CCS.
10:    Calculate the predicted queue Q_c^{𝒩_n} of the FAP 𝒩_n.
11:  end for
12:  The CCS update W_{t+1} according to Eq.(16).
13: end for
14: return The predicted popularity priority queue Q_c.
```

Federated Aggregation. After receiving the uploaded local models $\mathbf{H}_t$ from FAPs, the CCS updates the global model $\mathbf{W}_{t+1}$ (Line 12). To address the issue of imbalance in the local models across different FAPs, a weighted federated aggregation approach is employed by assigning different aggregation weights to

the local models uploaded by different FAPs. In that case, the updated global model $\mathbf{W}_{t+1}$ is:

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \sum_{\mathcal{N}_n \in \mathcal{N}} \nabla_t^{\mathcal{N}_n} \frac{M_t^{\mathcal{N}_n}}{\sum\limits_{\mathcal{N}_n \in \mathcal{N}} M_t^{\mathcal{N}_n}} (\mathbf{W}_t - \mathbf{H}_t^{\mathcal{N}_n}) \tag{16}$$

where $\nabla_t^{\mathcal{N}_n}$ represents the gradient step size and $M_t^{\mathcal{N}n}$ the number of service requests received by FAP $\mathcal{N}n$ at time t.

Algorithm 2. Cooperative Service Caching Mechanism

```
for n = 1, 2, ..., N do
  for the service requests received by N_n at time t do
    Calculate the service fee F_l and caching cost C^x_{n,l} for service l according to Eq.(3) and Eq.(4) respectively.
    if service l is cached in FAP N_n then
      Calculate L^{local}_{n,l} and E^{local}_{n,l} according to Eq.(5).
    end if
    if service l is cached in another FAP then
      Calculate L^m_{n,l} and E^m_{n,l} according to Eq.(7).
    end if
    if service l is not cached in any FAP then
      Calculate L^{ccs}_{n,l} and E^{ccs}_{n,l} according to Eq.(8).
    end if
    Calculate the average U_l of service l according to Eq.(11) and update t_l of service l according to Eq.(13a).
  end for
  Calculate the profit V_n for FAP-N according to Eq.(10).
end for
return Profits for each FAPs.
```

4.2 Deep Reinforcement Learning for Caching Decisions

Upon receiving the service popularity priority queue at each FAP for the current time slot, we utilize a deep Reinforcement learning model to determine the optimal cooperative caching decisions. The objective of this approach is to maximize the profits of FAPs while maintaining QoS for users.

State. We consider the services cached by FAP $\mathcal{N}_n$ as the current state $s^{\mathcal{N}_n}(t)$, where the cached services are primarily selected based on the predicted queue $Q_c^{\mathcal{N}_n}$ obtained from Algorithm 1. Therefore, the current state can be represented as $s(t)^{\mathcal{N}_n} = (s_1^{\mathcal{N}_n}, s_2^{\mathcal{N}_n}, ..., s_c^{\mathcal{N}_n})$, where $s_i^{\mathcal{N}_n}$ represents the ith popular service cached in FAP $\mathcal{N}_n$.

Algorithm 3. Deep Reinforcement Learning for Caching Decisions

Input: A set of service requests and the predicted popularity priority queue Q_c
Output: The caching decisions and profit of FAPs

for $t = 1, 2, \ldots, T$ **do**
  for $n = 1, 2, \ldots, N$ **do**
    Obtain the state $s(t)$
    Obtain the predicted popularity priority queue $Q_c^{\mathcal{N}_n}$.
    Calculate the action $a(t)$ according to Eq.(17).
    Obtain the next state $s(t+1)$ after executing $a(t)$.
    Obtain the profit according to Algorithm.(2).
    Obtain the reward $r(t)$.
    Store the tuple $(s(t), a(t), r(t), s(t+1))$ and randomly sample a minibatch from it.
    Calculate the loss function by Eq.(20).
    Calculate the gradient by Eq.(21).
    Update θ according to Eq.(22).
  end for
  Obtain the caching decisions according to θ.
  Each FAPs selects the services from the prediction queue $Q_c^{\mathcal{N}_N}$ for replacement.
end for
return The caching decisions and profit of FAPs.

Action. We define the action $a = (a^{\mathcal{N}_1}, a^{\mathcal{N}_2}, ..., a^{\mathcal{N}_n})$ to represent the set of actions for all FAPs.where $a^{\mathcal{N}_n} = (a_1^{\mathcal{N}_n}, a_2^{\mathcal{N}_n}, ..., a_c^{\mathcal{N}_n})$ represents whether it is necessary to replace the service in FAP $\mathcal{N}_n$. In this context, $a_i^{\mathcal{N}_n} = 0$ indicates that there is no need to replace the service stored in the ith position of FAP $\mathcal{N}_n$ cache, while $a_i^{\mathcal{N}_n} = 1$ indicates that it is necessary to replace the service stored in the ith position of the FAP $\mathcal{N}_n$. In case that the action function is implemented using the ε-greedy method:

$$a(t) = \text{argmax}(Q(s(t), a; \theta)) \tag{17}$$

Reward. We define the reward function r(t) to maximize the profit obtained by FAPs. After taking action $a(t)$, the corresponding reward $r(t)$ is obtained and the transition from state $s(t)$ to $s(t+1)$ occurs. Consequently, we can construct a $(s(t), a(t), r(t), s(t+1))$ transition, which is stored in the replay buffer. Then, the action-value function is updated:

$$Q(s_{i+1}, a_{i+1}; \theta) = Q(s_i, a_i; \theta) + \alpha [y_i - Q(s_i, a_i; \theta)] \tag{18}$$

where α represents the learning rate and y_i the target Q-value of the target network of tuple i:

$$y_i = r_i + \gamma \max \hat{Q}(s_{i+1}, a_{i+1}; \theta) \tag{19}$$

where γ is the discount factor. The loss function $L(\theta_i^n)$ of network is:

$$L(\theta_i^n) = \mathbb{E}\left[(y_i - Q(s_j, a_j; \theta))^2\right] \tag{20}$$

The gradient calculation of the loss function $\nabla_\theta L(\theta)$ for all sampled tuples is:

$$\nabla_\theta L(\theta) = \mathbb{E}\left[\left(y_i - Q\left(s_i, a_i, \theta\right)\right) \nabla_{\theta^i} Q\left(s_i, a_i, \theta\right)\right] \tag{21}$$

At the end of time slot t, the parameters of the network θ are updated as:

$$\theta \leftarrow \theta - \eta_\theta \nabla_\theta L(\theta) \tag{22}$$

where η_θ is the learning rate of prediction network.

Firstly, at each time instant t, we take the predicted popularity priority queue Q_c that obtained in Algorithm (1) and the cache state $s(t)$ of the FAPs as input (Line 3–4). Secondly, we use a deep reinforcement learning model that combines the profit calculation model in Algorithm (2) as the model indicator for training to obtain the optimal caching decisions model (Line 5–12). Finally, each FAP $\mathcal{N}_N$ selects the services from the prediction queue $Q_c^{\mathcal{N}_N}$ for replacement according to the caching decisions model (Line 14–15).

5 Performance Evaluation

5.1 Simulation Configuration

In this paper, we developed a simulation environment based on the Movielens dataset (ml–25 m) [22], which consist of 25,000,095 ratings and 1,093,360 tags of 62,423 movies created by 162,541 users. The datasets also include the related information about the involved movies, such as titles and genres, as well as user attributes including ID number, gender, age and postcode. We assumed that user preferences are represented by movie ratings and the number of ratings corresponds to the number of user preferences. The publication time of ratings is considered as the request initiation time. All the experiments are conducted on the same computer with an AMD Ryzen7 4800H 2.90 GHz processor, 16.0 GB of RAM and using PyTorch 2.0.

5.2 Baselines

We compare our method against four baselines:

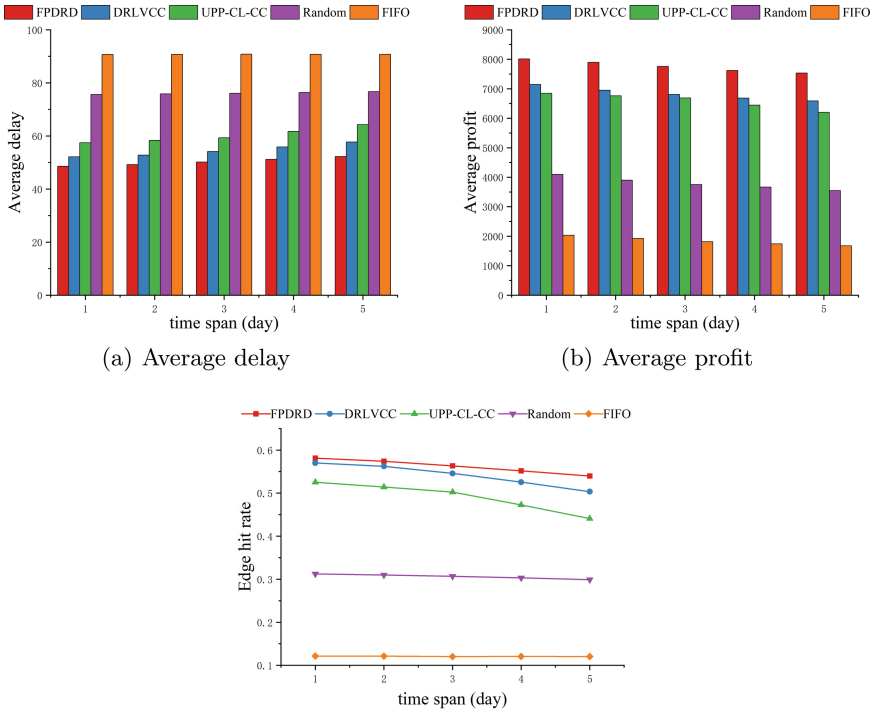
1) DRLVCC: The baseline initially employs a Convolutional Neural Network (CNN) model to assess the popularity of new requests at different locations. Subsequently, by a path-responsive vertical cooperative caching approach based on a deep reinforcement learning model to formulate caching decisions [23].
2) UPP-CL-CC: The baseline employs an LSTM model to dynamically capture user activities and preferences, thereby extracting local popularity information for FAPs which are subsequently subjected to clustering. Building upon this foundation, the author proposed a novel greedy approach to address the cache placement issue [24].

3) Random: Each FAP replaces unrequested services with a probability of ϵ. In our simulation, $\epsilon = 0.1$.
4) First-In-First-Out Scheme (FIFO): FAPs cache services based on the order of service requests and discard the oldest cached services when the cache space runs out.

5.3 Performance Analysis

We perform experiments under three scenarios:

1) We intercept different time spans of the datasets and increased them by one day at a time to observe how time spans impact service caching performance.
2) We study the impact of the number of service types on algorithm performance while fixing the time interval at 1 day and setting FAPs' cache capacity to 100.
3) We compare how FAPs' cache capacity influences algorithm performance while keeping the time interval fixed at 1 day and the number of service types fixed at 1000.



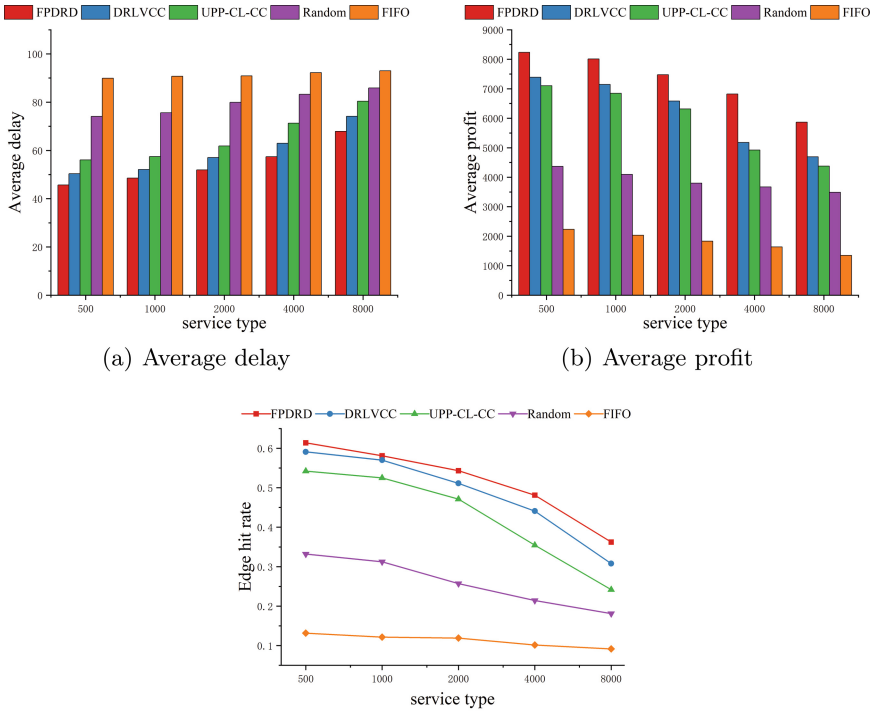
(a) Average delay

(b) Average profit

(c) Edge hit ratio

Fig. 3. The performance of algorithms in different time spans.

As shown in Fig. 3, the FPDRD method exhibits the best overall performance. Fig. 3(c) indicates that by considering the temporal variation of overall service popularity and the specific preferences of local users, our predictive queue aligns more closely with real-world scenarios, leading to significantly higher caching hit rates compared to the baseline algorithm. Figure 3(a) and Fig. 3(b) demonstrate that the FPDRD method achieves the lowest average delay and highest average profit on various time-span datasets. This achievement is attributed to the combination of a more accurate predictive model and the training of the decision-making network using DRL algorithms, resulting in a caching decisions model that can simultaneously safeguard the QoS of users and enhance providers' of network services' profit.



(a) Average delay

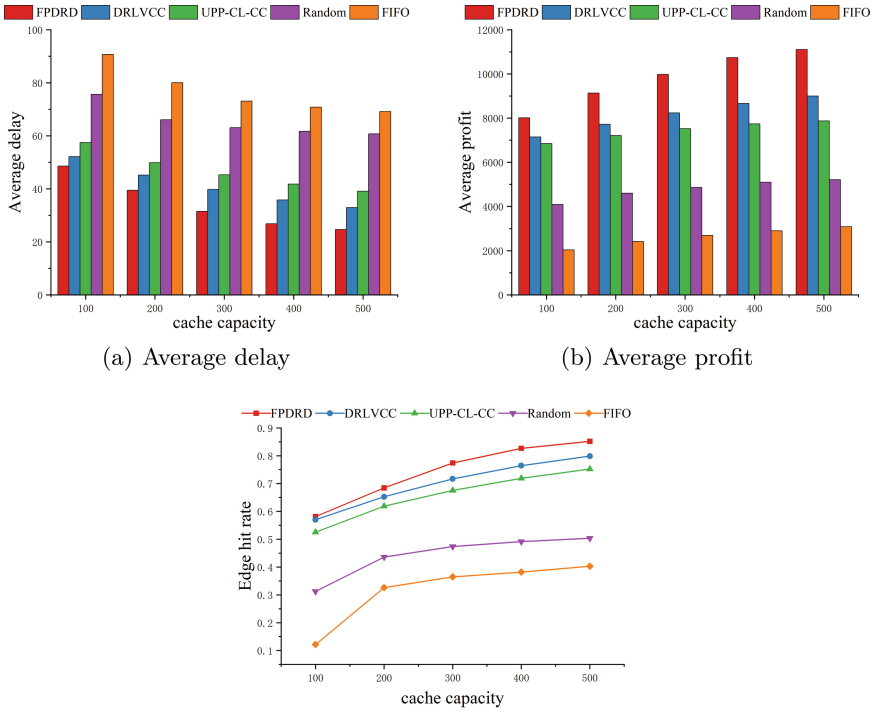
(b) Average profit

(c) Edge hit ratio

Fig. 4. The impact of the number of service types on algorithms.

As shown in Fig. 4, it is clear that an increase in service types negatively impacts the performance of all algorithms. However, different algorithms show different degrees of performance change. In particular, the FPDRD method exhibits a slower performance degradation while still maintaining the best overall performance. In contrast, the UPP-CL-CC method experiences a more rapid

performance degradation. The observed trends suggest a trade-off between the algorithms' capacity to adapt and optimize caching decisions effectively as the number of service types rises. The ability of the FPDRD method to respond quickly to environmental changes enables it to maintain superior overall performance even when confronted with a progressively diverse range of service types.



(a) Average delay

(b) Average profit

(c) Edge hit ratio

Fig. 5. The impact of the FAPs cache capacity on algorithms.

As shown in Fig. 5, the FPDRD algorithm achieves the highest performance for different FAPs cache capacities. Additionally, with increasing cache capacity of FAPs, the performance improvement of FPDRD method becomes more pronounced. The experimental results demonstrate the efficiency of the FPDRD method in popularity prediction and caching decisions, allowing for the efficient utilization of the available cache resources. As the cache capacity of FAPs increases, the algorithm can utilize this extra storage capacity to make more informed and optimized caching decisions. Therefore, the algorithm improves the cache hit rate and quality of service, ultimately enhancing network services and benefiting both end users and network service providers.

6 Conclusion

This paper investigates the service caching problem in MEC and proposes a novel caching method by leveraging a federated learning model for popularity prediction and a deep reinforcement learning model for yielding caching decisions (FPDRD). The experimental results clearly demonstrate that the superiority of the FPDRD method in achieving improved cache hit rates in MEC. This is accomplished by effectively considering the temporal variability of overall service popularity and the specificity of local user preference perception in predictions. Furthermore, the proposed method maximizes the utilization of limited storage and computational resources by promoting collaboration among FAPs. In case that the FPDRD method ensures the QoS of users and maximize the profits of network service providers. In the future, we aim to address the problem of resource idleness in FAPs due to the mismatch between storage and computing resource requirements of service and plan to optimize the fault-tolerance in collaborative service caching.

Acknowledgement. This work was supported in part by the Key Research and Development Project of Henan Province under Grant No. 231111211900, in part by the Henan Province Science and Technology Project under Grant No. 232102210024.

References

1. Wu, C., Peng, Q., Xia, Y., Jin, Y., Hu, Z.: Towards cost-effective and robust AI microservice deployment in edge computing environments. Futur. Gener. Comput. Syst. **141**, 129–142 (2023). https://doi.org/10.1016/j.future.2022.10.015
2. Hu, Q., Peng, Q., Shang, J., Li, Y., He, J.: EBA: an adaptive large neighborhood search-based approach for edge bandwidth allocation. In: Gao, H., Wang, X., Wei, W., Dagiuklas, T. (eds.) CollaborateCom 2022. LNICST, vol. 460, pp. 249–268. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-24383-7-14
3. Cruz, P., Achir, N., Viana, A.C.: On the edge of the deployment: a survey on multi-access edge computing. ACM Comput. Surv. **55**(5) (2022). https://doi.org/10.1145/3529758
4. Liu, G., et al.: An adaptive DNN inference acceleration framework with end-edge-cloud collaborative computing. Futur. Gener. Comput. Syst. **140**, 422–435 (2023). https://doi.org/10.1016/j.future.2022.10.033
5. Sharghivand, N., Derakhshan, F., Mashayekhy, L., Mohammadkhanli, L.: An edge computing matching framework with guaranteed quality of service. IEEE Trans. Cloud Comput. **10**(3), 1557–1570 (2022). https://doi.org/10.1109/TCC.2020.3005539
6. Huang, C.K., Shen, S.H.: Enabling service cache in edge clouds. ACM Trans. Internet Things **2**(3) (2021). https://doi.org/10.1145/3456564
7. Gao, J., Kuang, Z., Gao, J., Zhao, L.: Joint offloading scheduling and resource allocation in vehicular edge computing: a two layer solution. IEEE Trans. Veh. Technol. **72**(3), 3999–4009 (2023). https://doi.org/10.1109/TVT.2022.3220571
8. Liu, T., Zhang, Y., Zhu, Y., Tong, W., Yang, Y.: Online computation offloading and resource scheduling in mobile-edge computing. IEEE Internet Things J. **8**(8), 6649–6664 (2021). https://doi.org/10.1109/JIOT.2021.3051427

9. Xue, Z., Liu, C., Liao, C., Han, G., Sheng, Z.: Joint service caching and computation offloading scheme based on deep reinforcement learning in vehicular edge computing systems. IEEE Trans. Veh. Technol. **72**(5), 6709–6722 (2023). https://doi.org/10.1109/TVT.2023.3234336
10. Zong, T., Li, C., Lei, Y., Li, G., Cao, H., Liu, Y.: Cocktail edge caching: ride dynamic trends of content popularity with ensemble learning. IEEE/ACM Trans. Networking **31**(1), 208–219 (2023). https://doi.org/10.1109/TNET.2022.3193680
11. Li, T., Li, D., Xu, Y., Wang, X., Zhang, G.: Temporal-spatial collaborative mobile edge caching with user satisfaction awareness. IEEE Trans. Netw. Sci. Eng. **9**(5), 3643–3658 (2022). https://doi.org/10.1109/TNSE.2022.3188658
12. Li, Y., et al.: Collaborative content caching and task offloading in multi-access edge computing. IEEE Trans. Veh. Technol. **72**(4), 5367–5372 (2023). https://doi.org/10.1109/TVT.2022.3222596
13. Li, Z., Yang, C., Huang, X., Zeng, W., Xie, S.: Coor: collaborative task offloading and service caching replacement for vehicular edge computing networks. IEEE Trans. Veh. Technol. **72**(7), 9676–9681 (2023). https://doi.org/10.1109/TVT.2023.3244966
14. Xu, Z., et al.: Energy-aware collaborative service caching in a 5g-enabled MEC with uncertain payoffs. IEEE Trans. Commun. **70**(2), 1058–1071 (2022). https://doi.org/10.1109/TCOMM.2021.3125034
15. Lin, C.C., Chiang, Y., Wei, H.Y.: Collaborative edge caching with multiple virtual reality service providers using coalition games. In: 2023 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–6 (2023). https://doi.org/10.1109/WCNC55385.2023.10118763
16. Zhou, H., Zhang, Z., Li, D., Su, Z.: Joint optimization of computing offloading and service caching in edge computing-based smart grid. IEEE Trans. Cloud Comput. **11**(2), 1122–1132 (2023). https://doi.org/10.1109/TCC.2022.3163750
17. Ma, X., Zhou, A., Zhang, S., Wang, S.: Cooperative service caching and workload scheduling in mobile edge computing. In: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, pp. 2076–2085 (2020). https://doi.org/10.1109/INFOCOM41043.2020.9155455
18. Wu, R., Tang, G., Chen, T., Guo, D., Luo, L., Kang, W.: A profit-aware coalition game for cooperative content caching at the network edge. IEEE Internet Things J. **9**(2), 1361–1373 (2022). https://doi.org/10.1109/JIOT.2021.3087719
19. Xu, Z., et al.: Near-optimal and collaborative service caching in mobile edge clouds. IEEE Trans. Mob. Comput. **22**(7), 4070–4085 (2023). https://doi.org/10.1109/TMC.2022.3144175
20. Li, Y., Liang, W., Li, J.: Profit driven service provisioning in edge computing via deep reinforcement learning. IEEE Trans. Netw. Serv. Manage. **19**(3), 3006–3019 (2022). https://doi.org/10.1109/TNSM.2022.3159744
21. Wang, Z., Du, H.: Collaborative coalitions-based joint service caching and task offloading for edge networks. Theoret. Comput. Sci. **940**, 52–65 (2023). https://doi.org/10.1016/j.tcs.2022.10.037
22. Harper, F.M., Konstan, J.A.: The movielens datasets: history and context. ACM Trans. Interact. Intell. Syst. **5**(4) (2015). https://doi.org/10.1145/2827872
23. Liu, Y., Jia, J., Cai, J., Huang, T.: Deep reinforcement learning for reactive content caching with predicted content popularity in three-tier wireless networks. IEEE Trans. Netw. Serv. Manage. **20**(1), 486–501 (2023). https://doi.org/10.1109/TNSM.2022.3207994
24. Somesula, M.K., Rout, R.R., Somayajulu, D.: Greedy cooperative cache placement for mobile edge networks with user preferences prediction and adaptive clustering. Ad Hoc Netw. **140**, 103051 (2023). https://doi.org/10.1016/j.adhoc.2022.103051