Federated Deep Reinforcement Learning for Recommendation-Enabled Edge Caching in Mobile Edge-Cloud Computing Networks

Chuan Sun, *Student Member, IEEE*, Xiuhua Li, *Member, IEEE*, Junhao Wen, *Member, IEEE*, Xiaofei Wang, *Senior Member, IEEE*, Zhu Han, *Fellow, IEEE*, and Victor C. M. Leung, *Life Fellow, IEEE*

***Abstract*— To support rapidly increasing services and applications from users, multi-tier computing is emerged as a promising system-level computing architecture by distributing computing/caching/communication/networking capabilities between cloud servers to users, especially deploying edge servers at network edges (e.g., base stations). However, due to heterogeneous content requests of users and a high-cost hit manner with direct hits, edge caching is still a most serious issue to be addressed. In this paper, we investigate the issue of recommendation-enabled edge caching in mobile two-tier (edge-cloud) computing networks. Particularly, we integrate recommender systems and edge caching to support both direct hits and soft hits and thus improve the resource utilization of edge servers. We model the factors affecting the user quality of experience as a comprehensive system cost and further formulate the problem as a multi-agent Markov decision process with the goal of minimizing the long-term average system cost. To address the formulated problem, we propose a decentralized recommendation-enabled edge caching framework that leverages a discrete multi-agent variant of soft actor-critic and federated learning. The proposed framework enables each edge server to learn its best policy locally and generate judicious decisions independently. Finally, trace-driven simulation results demonstrate that the proposed framework converges to a better caching policy and outperforms several existing algorithms on average system cost reduction.**

***Index Terms*— Multi-tier computing, recommendation-enabled edge caching, soft hits, federated learning, discrete soft actor-critic.**

I. Introduction

CURRENTLY, massive end devices access the Internet for realizing various services and applications (such as content access, virtual navigation/management, and environmental monitoring) in wireless networks, which unavoidably leads to serious backbone network congestion and degrades quality of service/experience (QoS/QoE) of users [1], [2], [3]. According to the Ericsson mobility report [4], around 230 million end devices will generate 288 ExaBytes' data per month in 2027. To address these challenges, multi-tier computing has been emerged as a promising system-level computing architecture by endowing network edges (e.g., base stations (BSs)) with computing/caching functionalities [5], [6], [7]. In next generation wireless networks, BSs deployed with edge servers (e.g., Nvidia Jetson TX2[1]) can cache some contents in proximity to users for reducing the retrieving latency and alleviating network congestion, called *edge caching* [8], [9].

Compared to cloud servers, edge servers can only cache a tiny fraction of content catalogs due to their limited cache capacity [10]. Besides, users' content requests and network conditions are dynamically changing, which leads to massive request misses, i.e., the requested contents are not cached at edge servers. Hence, edge caching highly relies on the scheme design of cache replacement to enhance the cache hit ratio and decrease service latency [11]. Some rules-based cache replacement policies have been widely used, such as Least Recently Used (LRU) [12] and Least Frequently Used (LFU) [13]. These conventional policies cannot cope with large-scale complex situations. Recently, lots of cache replacement policies based on deep learning have been presented for improving QoS/QoE [14]. Especially, the reinforcement learning based cache replacement policy is representative,

Manuscript received 15 April 2022; revised 2 September 2022; accepted 16 November 2022. Date of publication 12 January 2023; date of current version 16 February 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFE0125400; in part by the National NSFC under Grant 61902044, Grant 62072060, Grant 62072332, and Grant 62102053; in part by the Graduate Research and Innovation Foundation of Chongqing under Grant CYB21068; in part by the Chongqing Research Program of Basic Research and Frontier Technology under Grant cstc2022ycjh-bgzxm0058; in part by the Key Research Program of Chongqing Science and Technology Commission under Grant cstc2021jscx-dxwtBX0019; in part by the Haihe Laboratory of ITAI under Grant 22HHXCJC00002; in part by the Guangdong Pearl River Talent Recruitment Program under Grant 2019ZT08×603 and 2019JC01×235; and in part by NSF under Grant CNS-2107216, Grant CNS-2128368, and Grant CMMI-2222810. *(Corresponding author: Xiuhua Li.)*

Chuan Sun and Junhao Wen are with the School of Big Data and Software Engineering, Chongqing University, Chongqing 401331, China (e-mail: c.sun@cqu.edu.cn; jhwen@cqu.edu.cn).

Xiuhua Li is with the School of Big Data and Software Engineering, Chongqing University, Chongqing 401331, China, and also with the Haihe Laboratory, ITAI, Tianjin 300072, China (e-mail: lixiuhua1988@gmail.com).

Xiaofei Wang is with the College of Intelligence and Computing, Tianjin University, Tianjin 300072, China (e-mail: xiaofeiwang@tju.edu.cn).

Zhu Han is with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004 USA, and also with the Department of Computer Science and Engineering, Kyung Hee University, Seoul 446-701, South Korea (e-mail: hanzhu22@gmail.com).

Victor C. M. Leung is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China, and also with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC V6T1Z4, Canada (e-mail: vleung@ieee.org).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2023.3235443.

Digital Object Identifier 10.1109/JSAC.2023.3235443

[1]https://developer.nvidia.com/embedded/jetson-tx2

which has a learning ability to sense and capture large-scale user requests in dynamic networks [11], [15].

However, current edge caching in multi-tier computing networks still faces two serious challenges: *a high-cost hit manner with direct hits* and *heterogeneous content requests of users*. On the one hand, current request processing is in a manner of direct hits, i.e., the requested contents must be retrieved no matter how many costs (e.g., transmission latency) are taken [16]. For instance, a user wants to watch a movie by smartphone. But the movie is not cached at the local edge server and can only be fetched from the remote cloud server. Since the movie is very bandwidth-consuming, this transmission process takes a longer latency cost. This user may suffer from poor QoE, especially during rush hours on Internet [17]. As edge servers cache a tiny fraction of content catalogs, direct hits cannot meet massive users' requests and cause a low hit ratio. Therefore, direct hits generally take massive costs, e.g., latency, energy consumption, and payment fee, when facing certain tricky objective situations (e.g., user request miss, backbone network congestion, and copyright restriction). On the other hand, the effectiveness of cache replacement policies is severely affected by users' content requests [15]. More specifically, the cache hit ratio increases only when the cached contents at edge servers are requested by multiple users, i.e., users' homogeneous content requests. However, different users' content requests are generally highly heterogeneous and the distributions among content requests satisfied by different edge servers are skewed [18], [19]. Therefore, the same cache replacement policy for all edge servers will be dramatically defective. To this end, two challenges are summarized as:

- *Request level:* the request heterogeneity degrades the cache replacement policy.
- *Service level:* direct hits cannot meet massive users' requests and take massive costs.

To address the above challenges, combining recommender systems and edge caching is considered as an effective method to improve caching gains, e.g., transmission reliability, cache hit ratio, or effective throughput [16], [20], [21], [22], [23], [24]. Since recommender systems not only meticulously discover the personalized user preference on contents but also reshape the content request schema of users, recommender systems have greatly/extremely affected users' behavior on the Internet surfing (e.g., YouTube, Tiktok, Bilibili) [25]. It is reported that up to 80% of requests on popular content distribution platforms come from user recommendations [26]. To satisfy the user herself/himself rather than each specific content request, soft hits based on recommender systems have been proposed by recommending alternative/inter-changeable contents. Specifically, when a user faces the above mentioned tricky situations, other highly related or immensely similar contents (e.g., a recent FIFA game and the same uploader's cat clip) can be recommended to this user, i.e., substituting goods in microeconomic [27]. Soft hits with a higher cache hit ratio not only reduce the transmission latency and network cost by delivering local contents to users, but also effectively improve the resource utilization of edge servers.

In this paper, we investigate the issue of recommendation-enabled edge caching in mobile edge-cloud computing networks, which jointly considers the challenges of the resource level and the decision level. Edge-cloud computing is a case of two-tier computing, which collaborates the edge layer and the cloud layer. In particular, the proposed system integrates recommender systems to support both direct hits and soft hits. Therefore, the improved resource utilization of edge servers can effectively make up the limits of cache capacity. To alleviate the effects of soft hits on the user QoE, we model the user QoE as a comprehensive system cost. To address the formulated problem, we propose a decentralized caching algorithm with joint deep reinforcement learning (DRL) and federated learning (FL), where multiple agents learn and make decisions independently. Specifically, our main contributions are summarized as follows.

- To improve caching gains, we integrate edge caching and recommender systems to form a recommendation-enabled edge caching system in mobile edge-cloud computing networks. The proposed system supports both direct hits and soft hits. We model the factors affecting the user QoE as a comprehensive system cost (consisting of similarity cost, latency cost, and cache hit cost). We further formulate the cache replacement problem as a multi-agent Markov decision process (MDP) to minimize the expected long-term system cost (reflecting the user QoE).
- We decompose the formulated problem into two simpler subproblems and propose a decentralized recommendation-enabled edge caching framework. Particularly, to deal with the heterogeneity, we propose a federated discrete soft actor-critic (FDSAC) algorithm that only federates the critic network for alleviating the request heterogeneity. Besides, we improve the actor network by integrating long short term memory (LSTM) to capture time-series dependencies of replacement actions.
- We perform extensive simulations over a synthetic dataset and a MovieLens dataset. Trace-driven simulation results show that the proposed framework outperforms several existing rule-based and DRL-based algorithms on convergence. Moreover, it can significantly reduce the average system cost for improving the user QoE, especially the average latency cost and the average cache hit cost.

The rest of this paper is organized as follows. Section II summarizes the related work. Sections III and IV introduce the system model and the decentralized recommendation-enabled edge caching framework, respectively. We evaluate the performance of the proposed framework in Section V. Conclusion is given in Section VI.

II. Related Work

A. Convergence of Edge Caching and Recommender Systems

Combining edge caching and recommender systems differs from conventional recommendation algorithms in purpose, manner, and scenario. Existing studies mainly include two research directions: predicting the probability of request transition and increasing the recommendation decision. For the first

TABLE I
IMPORTANT NOTATIONS

Notation	Definition
$\mathcal{B}$	Set of BSs/edge servers
$\mathcal{M}$	Set of end devices
$\mathcal{F}$	Set of contents
$\mathcal{T}$	Set of time slots
$\mathcal{P}$	Set of requests
K_b	Cache size of edge servers b
κ_f	Size of content f
$\iota_{m,b}^t$	Association between end device m and BS b
ν_b^t	Similarity cost of BS b in time slot t
ξ_b^t	Latency cost of BS b in time slot t
μ_b^t	Cache hit cost of BS b in time slot t
$\mathcal{S}_b^t$	State of BS b in time slot t
$\mathcal{A}_b^t$	Action of BS b in time slot t
r_b^t	Negative scalar reward of BS b in time slot t
π_b	Policy of BS b

direction, the authors in [20], [21], and [28] investigated the issue of user request transition matrices and proposed a MDP formulation to predict recommendation-enabled user requests. They mainly focused on user behavior mining via recommendation algorithms, but did not consider the issue of edge cache replacement. For another direction, the studies in [16], [29], and [23] investigated the joint optimization problem of making cache replacement and recommendation decisions to maximize the cache hit ratio. The authors in [30] proposed a novel area of similarity caching networks, where requests could be forwarded along a path of caches to get the best efficiency-accuracy tradeoff. But these studies considered a simple scenario with only one user in the proposed system. To address this problem, the studies in [31] and [32] further considered joint caching and recommendation in multi-user mobile edge computing networks and proposed polynomial-time algorithms to maximize the weighted sum of the streaming quality and the recommendation quality. However, these studies utilized conventional optimization algorithms to address one-term caching optimization problems, which are not suitable for complex scenarios with long-term optimization. Besides, existing studies consider the effects of soft hits and lack a comprehensive metric to reflect the user QoE.

B. DRL-Based Cache Replacement

Designing cache replacement policies are mainly driven by content popularity. Some conventional rule-based algorithms (e.g., LRU [12] and LFU [13]) can only roughly predict content popularity to make ordinary cache replacement decisions. The studies in [33] proposed a cache replacement policy for achieving the trade-off between multihop communication costs and the freshness of transient data items. These conventional algorithms cannot cope with dynamic and large-scale users' requests and complex mobile edge network environments. Recently, massive learning-based algorithms (e.g., deep learning and DRL) have been regarded as promising techniques and widely applied in edge caching [34], [35], [36], [37]. Sun et al. [34] proposed the use of deep reinforcement learning to predict the location of the unmanned aerial vehicles (UAVs) and the contents to cache at the UAVs. To maximize the cache hit ratio and minimize the transmission delay, the authors in [35] proposed two deep actor-critic (AC) reinforcement learning-based policies, i.e., centralized and decentralized. To deal with dynamic environment, Wang et al. [36] proposed a federated deep-reinforcement-learning-based cooperative edge caching algorithm. DRL algorithms in prior studies, e.g., AC and DQN, suffer from high computational complexity and brittleness to scalability. To alleviate these bottlenecks, Wu et al. [38] proposed a novel DRL approach based on SAC for discrete space. But this DRL algorithm will take additional transmission cost due to frequent parameter transmissions between the cloud server and edge servers during the process of training learning models. Most existing studies about cache replacement ignore the issue of request heterogeneity.

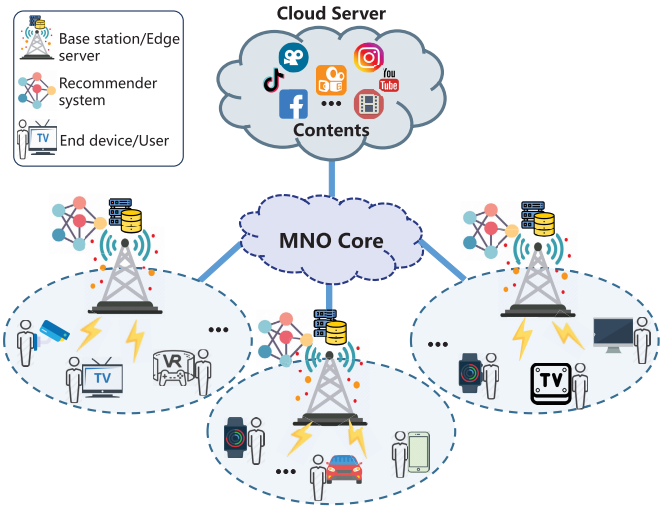


Fig. 1. Illustration of recommendation-enabled edge caching architecture in a mobile edge-cloud computing network.

III. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we introduce the system model of recommendation-enabled edge caching and the problem formulation. Specifically, Sec. III-A presents the recommendation-enabled edge caching architecture. Sec. III-B introduces how to recommend contents. We model a comprehensive system cost to reflect the user QoE in Sec. III-C. Then we introduce the cache replacement model in Sec. III-D. Finally, we formulate the optimization problem in Sec. III-E. Some key notations are summarized in Table I.

A. Recommendation-Enabled Edge Caching Architecture

As illustrated in Fig. 1, we consider a mobile edge-cloud computing network, which is a case of two-tier computing networks. In the considered network, computing and caching services are brought to the network edge (e.g., BSs) from the cloud server or mobile network operator (MNO). Hence, the edge servers deployed at BSs can cache finite contents from the cloud server through backhaul links.[2] Meanwhile, the edge servers support direct hits and soft hits based on recommender

[2]As fibers are considered as ideal backhaul media to provide sufficient bandwidth as well as future-proof capacity upgrade, backhaul links are assumed to be fibers.

systems. Numerous users/end devices[3] are geographically distributed in the served cellular area of BSs. The end devices in each cell are associated with the local BS via wireless cellular links. We consider the scenario with heterogeneous content requests of users, so that, the distributions of the cached contents among edge servers are diverse. The cooperation of edge servers via using BS-BS links may not be cost-effective due to two practical aspects: i) deploying and operating BS-BS links is generally expensive from the economic perspective; ii) fetching contents among different edge servers only achieves limited caching gains, but brings high complexity of content management. Thus, to reduce the system cost and content management complexity, we assume that each user requests the desired contents either locally from the served edge server or directly from the cloud server.

In the considered network, a total of B BSs (denoted as $\mathcal{B} = \{1, 2, \ldots, B\}$) are connected to the cloud server. Here, BS $b \in \mathcal{B}$ is equipped with an edge server with a limited cache size $K_b \in \mathbb{R}^+$. A total of F contents (denote as $\mathcal{F} = \{1, 2, \ldots, F\}$) can be requested by users. Denote $\kappa_f \in \mathbb{R}^+, \forall f \in \mathcal{F}$ as the size of content f. Generally, we assume that the cloud server has sufficient capacity to store all F contents. We denote a set of cached contents at edge server b as $\mathcal{F}_b$ (please see the details in Sec. III-C). Thus, edge server b meets the constraint $\sum_{f \in \mathcal{F}_b} \kappa_f \leq K_b$ due to the limit of cache capacity. We consider that a total of M end devices (denoted as $\mathcal{M} = \{1, 2, \ldots, M\}$) are randomly distributed in the service area. In the considered network, we focus on one episode that contains a set of time slots $\mathcal{T} = \{1, 2, \ldots, T\}$. Particularly, we consider that users keep moving, which leads to the changing user association with BSs. Let $\iota^t_{m,b} \in \{0, 1\}$ denote the association between user m and edge server b in the time slot $t \in \mathcal{T}$, where $\iota^t_{m,b} = 1$ represents that user m is served by edge server b, otherwise $\iota^t_{m,b} = 0$. We assume that each user can only be served by one adjacent BS (i.e., $\sum_{b \in \mathcal{B}} \iota^t_{m,b} = 1$) and the association will not change within each time slot. We further denote the set of users associated with BS b in time slot t as $\mathcal{M}^t_b = \{m \in \mathcal{M} | \iota^t_{m,b} = 1\} \subseteq \mathcal{M}$.

At the beginning of time slot $t \in \mathcal{T}$, user $m \in \mathcal{M}$ requests only one content $\rho^t_{m,f} \in \{0, 1\}$ from catalog $\mathcal{F}$, where $\rho^t_{m,f} = 1$ represents that user m requests content f, otherwise $\rho^t_{m,f} = 0$. We have the constraint $\sum_{f \in \mathcal{F}} \rho^t_{m,f} = 1$. There are several common request policies: offline, adversarial, and stochastic [22], where offline and adversarial are not suitable for edge caching. Generally, we consider a realistic scenario that users' content requests are stochastic and approximately follow the Zipf distribution [39]. Denote $\mathcal{P} = \{\rho^t_{m,f}\}_{m \in \mathcal{M}, f \in \mathcal{F}, t \in \mathcal{T}}$ as the set of all requests, which follows $\mathcal{P} \sim Zipf(\alpha, |\mathcal{F}|)$, where α is a parameter.

B. Content Recommendation Model

In this subsection, we present a content recommendation model to realize soft hits. The basic idea is that each user has the same preference for similar contents as shown in Fact 1. Once edge servers do not cache the requested contents but some similar and user-interested contents, the similar contents are recommended to users for satisfying their requests.

Fact 1: Two different contents have some potential similarities for a user [40].

To illustrate soft hits clearly, we consider the request process of a user (say Joy) as shown in Fig. 2. Joy initiates a request for content 6 as shown in Fig. 2(a). However, at this moment, the served edge server does not cache content 6 but caches some similar contents (e.g., content 1) for faster and cheaper access. There are two possible options for Joy to choose from: i) *Direct hit*, Joy insists on fetching content 6 from the cloud server with even more transmission latency as shown in Fig. 2(b); ii) *Soft hit*, Joy accepts similar contents for faster and cheaper access as shown in Fig. 2(c), and content 1 will be retrieved from the edge server to replace content 6.

Definition 1: Direct hits represent that users retrieve the requested contents; *Soft hits* represent that users retrieve another recommended contents.

The crux of soft hits is to discover the similarities of contents about users. As different users perform different behavior habits and content performances, each user has personalized similarities for contents. We assume that any content can be represented as a point in the d-dimensional Euclidean space $\mathbb{R}^d$. Therefore, we denote the similarity of any two contents $f, j \in \mathcal{F}$ for user $m \in \mathcal{M}$ as a non-negative score $\phi^m_{f,j}$ (please see the details in Sec. IV-B). To ease the calculation, we normalize $\phi^m_{f,j}$ from 0 to 1 ($\phi^m_{f,j} \in [0, 1]$). The similarity score of the content itself should be 1 ($\phi^m_{f,f} = 1$). We further denote the similarity matrix as $\Phi^m_{F \times F}$, which represents the set of similarity score $\phi^m_{f,j}$ and is symmetric. As catalog $\mathcal{F}$ is finite, edge servers can learn each user's content performances to build a similarity matrix $\Phi^m_{F \times F}$ via an advanced recommendation algorithm. We can recommend some similar contents to the user according to this similarity matrix. To this end, we have the following lemma to summarize this content recommendation model.

Lemma 1: When user $m \in \mathcal{M}$ initiates a request $\rho^t_{m,f} = 1$ for content $f \in \mathcal{F}$ at the beginning of time slot $t \in \mathcal{T}$, if edge server $b \in \mathcal{B}$ ($\iota^t_{m,b} = 1$) does not cache content f but caches similar content $j \in \mathcal{F}$, then user m may choose soft hits for content j with a probability (associated with $\phi^m_{f,j}$) rather than direct hits.

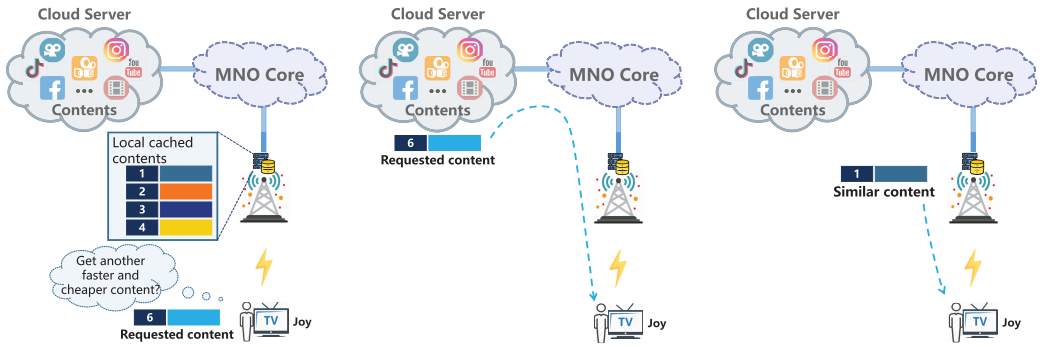
Proof: Please see Appendix A. ■

When the requested contents are not cached at edge servers, how to determine direct hits or soft hits remains a challenge. Soft hits can dramatically reduce the transmission latency by recommending the cached contents with high similarity scores. But direct hits are still chosen when the cached contents with low similarity scores. This is because we focus on improving the user QoE, not just reducing the transmission latency. Therefore, we model the factors affecting the user QoE as a comprehensive system cost.

C. System Cost Model

As the proposed system supports both direct hits and soft hits, users expect to fetch the requested contents with high similarities and low transmission latency. Besides, a low cache

[3]In this paper, "user" and "end device" are used interchangeably for convenience, so are "BSs" and "edge servers".



(a) Initiation of a request (b) Direct hits (c) Soft hits

Fig. 2. Illustration of the request process that Joy requests a content: (a) Joy initiates a request for content 6 not cached at the edge server; (b) Joy has to retrieve content 6 directly from the cloud server (Direct hits); (c) Joy accepts similar content 1 that can be retrieved from the edge server to replace content 6 (Soft hits).

hit ratio means that lots of users' content requests are fetched from the cloud server, which leads to backbone network congestion. Thus, we model a novel comprehensive system cost to reflect the user QoE, including similarity cost, latency cost, and cache hit cost.

1) Similarity cost: As mentioned above, similarity matrices represent the set of similarity scores for any two contents, which are calculated by recommender systems. Correspondingly, we denote the similarity cost for user $m \in \mathcal{M}$ as a non-negative cost $c^m(f,j), \forall f,j \in \mathcal{F}$ to approximate content f with content j. Generally, the similarity cost and the similarity score of two contents are interrelated, i.e., the higher the similarity score, the lower the similarity cost. As the similarity score meets the constraint $\phi^m_{f,j} \in [0,1]$, we model the similarity cost $c^m(f,j) = 1-\phi^m_{f,j}$, where $c^m(f,j) \in [0,1]$. Particularly, the similarity cost of the content itself should be 0 ($c^m(f,f) = 0$).

To determine direct hits or soft hits, a requested content should be compared with a set of cached contents at the edge server. We need to introduce the similarity cost between a content and a set of contents. We denote the local cache state of BS $b \in \mathcal{B}$ as $\boldsymbol{s}_b = (s_{b,f})_{f\in\mathcal{F}}$, where $s_{b,f} \in \{0,1\}$ and $s_{b,f} = 1$ represents that content f has been cached at edge server b, otherwise $s_{b,f} = 0$. As mentioned above, a set of cached contents $\mathcal{F}_b$ at edge server b are further expressed as $\mathcal{F}_b = \{f \in \mathcal{F} | s_{b,f} = 1\} \subseteq \mathcal{F}$. Denote the similarity cost between a content f and a set of cached contents $\mathcal{F}_b$ as $c^m(f,\mathcal{F}_b)$. As we focus on improving the user QoE, the content with the smallest similarity cost should be considered, i.e., $c^m(f,\mathcal{F}_b) = \min_{j\in\mathcal{F}_b} c^m(f,j)$.

2) Latency cost: There exist two types of transmission latency for fetching contents (i.e., cloud-edge latency and edge-end latency). The cloud-edge latency $l^C_f = \kappa_f / v_c$ represents that content $f \in \mathcal{F}$ is transmitted from the cloud server to edge servers through backhaul links [9], where v_c is the average transmission rate. The edge-end latency $l^E_{b,m,f}$ represents that content f is transmitted from BS $b \in \mathcal{B}$ to end device $m \in \mathcal{M}$ through wireless cellular links. We denote the downlink rate $v^t_{b,m}$ between end device m and BS b as follows

$$v^t_{b,m} = w^t_{b,m} \log_2\left(1 + \frac{p_b |h_{b,m}|^2}{\sigma^2}\right), \forall m \in \mathcal{M}^t_b, \quad \forall b \in \mathcal{B}, \forall t \in \mathcal{T}, \tag{1}$$

where σ^2 is the noise power; p_b is the transmit power allocated from BS b to its served users by employing an equal power allocation scheme [41], and the related issue of power allocation optimization is out of the scope of this paper and will be studied in future work; $h_{b,m}$ is the channel gain; $\boldsymbol{w}^t_b = (w^t_{b,m})_{m\in\mathcal{M}^t_b}$ are the allocated radio bandwidth resource, which meet the constraint $\sum_{m\in\mathcal{M}^t_b} w^t_{b,m} \le W_b$, where W_b is the total channel bandwidth. Therefore, the edge-end latency can be calculated by $l^E_{b,m,f} = \kappa_f / v^t_{b,m}$.

We denote $\boldsymbol{z}^t_b = (z^{E,t}_m, z^{C,t}_m, z^{S,t}_m)_{m\in\mathcal{M}^t_b}, \forall b \in \mathcal{B}, \forall t \in \mathcal{T}$ as the request processing action to meet all content requests of users, where $z^{E,t}_m, z^{C,t}_m, z^{S,t}_m \in \{0,1\}$ represent that the requested content of user m is offered by the edge server, the cloud server, and the edge server by a similar content, respectively. Particularly, we consider that contents are inseparable and users' content requests can be satisfied by only one place. Thus, the request processing action should meet the constraint $z^{E,t}_m + z^{C,t}_m + z^{S,t}_m = 1$. Hence, for BS b, we calculate the request processing latency $\mathcal{L}^t_b$ as

$$\mathcal{L}^t_b = \sum_{m\in\mathcal{M}^t_b} \sum_{f\in\mathcal{F}} \rho^t_{m,f} \left(\left(z^{E,t}_m + z^{C,t}_m\right) l^E_{b,m,f} + z^{C,t}_m l^C_f + z^{S,t}_m l^E_{b,m,j} \right), j = \arg\min_{j\in\mathcal{F}^t_b} c^m(f,j), \ \forall b \in \mathcal{B}, \forall t \in \mathcal{T}, \tag{2}$$

where $z^{S,t}_m l^E_{b,m,j}$ represents that the most similar content j is recommended to user m, and content j is obtained by $\arg\min_{j\in\mathcal{F}^t_b} c^m(f,j)$.

3) Cache hit cost: Cache hits represent that users' content requests can be offered by edge servers. As mentioned above, $z^{E,t}_m = 1$ or $z^{S,t}_m = 1$ represent that edge server b can offer the requested content or a similar content to user m. We denote the cache hit ratio $u^t_b \in [0,1]$ as the ratio of the sum of cache hits and the number of all requests, expressed as

$$u^t_b = \frac{\sum_{m\in\mathcal{M}^t_b} \sum_{f\in\mathcal{F}} \rho^t_{m,f} \left(z^{E,t}_m + z^{S,t}_m\right)}{|\mathcal{M}^t_b|}, \forall b \in \mathcal{B}, \quad \forall t \in \mathcal{T}, \tag{3}$$

where $|\mathcal{M}^t_b|$ denotes the number of end devices served by BS b. The higher cache hit ratio has the fewer effects on the user QoE, otherwise. Thus, we model the cache hit cost as $\mu^t_b = (1 - u^t_b) \in [0,1]$, which represents that the cache hit cost and the cache hit ratio are negative correlation.

D. Cache Replacement Model

Edge servers should replace the cached contents to hit more content requests of users. We model this cache replacement for edge server $b \in \mathcal{B}$ as an MDP [36], which consists of state, action, and reward.

1) State: As each edge server makes the replacement action by observing itself and users in the served cellular area, we consider that the state consists of the local cache state $\boldsymbol{s}_b^t$ and the request state in each time slot. Denote the request state $\boldsymbol{n}_b^t = (n_{b,f}^t)_{f\in\mathcal{F}}$ as a frequency vector, where $n_{b,f}^t = \sum_{m\in\mathcal{M}_b^t} \rho_{m,f}^t$ is the number of content f requested by all users at BS b. Therefore, we denote the state $\mathcal{S}_b^t$ as the combination of $\boldsymbol{s}_b^t$ and $\boldsymbol{n}_b^t$, represented as

$$\mathcal{S}_b^t = \left((s_{b,f}^t)_{f\in\mathcal{F}}, (n_{b,f}^t)_{f\in\mathcal{F}}\right), \forall b \in \mathcal{B}, \quad \forall t \in \mathcal{T}, \tag{4}$$

which is a $2F$-dimensional tuple.

2) Action: Edge servers should determine which contents are replaced and how content requests are met, i.e., the cache replacement action and the request processing action $\boldsymbol{z}_b^t$. Similar to [38], each edge server is assumed to select at most one content from catalog $\mathcal{F}$ to replace in each time slot. Hence, let $\boldsymbol{a}_b^t = (a_{b,f}^t)_{f\in\{0\}\cup\mathcal{F}}$ denote the cache replacement action of BS b, where $a_{b,f}^t \in \{0,1\}$ and $a_{b,f}^t = 1$ means content f should be cached, otherwise $a_{b,f}^t = 0$. Further, we have the constraint $\sum_{f\in\{0\}\cup\mathcal{F}} a_{b,f}^t = 1$. Particularly, if $f = 0$ and $a_{b,0}^t = 1$ represent that BS b does not replace any content in time slot t. Therefore, we denote the action $\mathcal{A}_b^t$ as the combination of $\boldsymbol{a}_b^t$ and $\boldsymbol{z}_b^t$, expressed as

$$\mathcal{A}_b^t = \left((a_{b,f}^t)_{f\in\{0\}\cup\mathcal{F}}, \boldsymbol{z}_b^t\right), \quad \forall b \in \mathcal{B}, \forall t \in \mathcal{T}, \tag{5}$$

which is a $(1 + F + 3|\mathcal{M}_b^t|)$-dimensional tuple.

3) Reward: According to the system cost model, we define the reward as the weighted sum of the similarity cost, the latency cost, and the cache hit cost, calculated as

$$\mathcal{R}_b^t(\mathcal{S}_b^t, \mathcal{A}_b^t) = \lambda_1\mu_b^t + \lambda_2\nu_b^t + \lambda_3\xi_b^t, \forall b \in \mathcal{B}, \quad \forall t \in \mathcal{T}, \tag{6}$$

where $\nu_b^t = \sum_{m\in\mathcal{M}_b^t}\sum_{f\in\mathcal{F}} \rho_{m,f}^t z_m^{S,t} c^m(f, \mathcal{F}_b^t)$ is the similarity cost when soft hits occur; $\xi_b^t = \sum_{f\in\{0\}\cup\mathcal{F}} a_{b,f}^t l_f^C + \mathcal{L}_b^t$ is the total transmission latency; $\lambda_1, \lambda_2, \lambda_3$ are weights and meet $\lambda_1 + \lambda_2 + \lambda_3 = 1$. As $\mu_b^t, \nu_b^t, \xi_b^t$ are all non-negative, the reward should meet $\mathcal{R}_b^t(\mathcal{S}_b^t, \mathcal{A}_b^t) \geq 0$.

E. Problem Formulation

In this paper, we focus on improving the user QoE by minimizing the system cost within one episode. We need to design a policy for each BS (regarded as an agent) to generate the optimal action. The policy π_b of BS $b \in \mathcal{B}$ is a mapping from its state to its action, i.e., $\pi_b : \mathcal{S} \to \mathcal{A}$. Let $\gamma \in (0, 1]$ denote the discount factor. To ease the calculation, we denote the negative scalar reward as $r_b^t = -\mathcal{R}_b^t(\mathcal{S}_b^t, \mathcal{A}_b^t)$. Thus, we aim to learn the optimal policy π_b^* via maximizing the expected long-term discounted cumulative negative reward, i.e.,

$$\max_{\pi_b} \mathbb{E}\left[\sum_{i=t}^{\infty} \gamma^{i-t} r_b^i \mid \pi_b\right] \tag{7a}$$

$$\text{s.t.} \quad \sum_{m\in\mathcal{M}_b^t} w_{b,m}^t \leq W_b, \sum_{f\in\mathcal{F}_b} \kappa_f \leq K_b, \quad \forall t \in \mathcal{T}, \tag{7b}$$

$$\sum_{f\in\{0\}\cup\mathcal{F}} a_{b,f}^t = 1, \quad \forall t \in \mathcal{T}, \tag{7c}$$

$$z_m^{E,t} + z_m^{C,t} + z_m^{S,t} = 1, \quad \forall m \in \mathcal{M}_b^t, \forall t \in \mathcal{T}, \tag{7d}$$

$$w_{b,m}^t \in [0, 1], \quad \forall m \in \mathcal{M}_b^t, \forall t \in \mathcal{T}, \tag{7e}$$

$$z_m^{E,t}, z_m^{C,t}, z_m^{S,t} \in \{0, 1\}, \forall m \in \mathcal{M}_b^t, \quad \forall t \in \mathcal{T}, \tag{7f}$$

$$a_{b,f}^t \in \{0, 1\}, \quad \forall f \in \{0\} \cup \mathcal{F}, \quad \forall t \in \mathcal{T}, \tag{7g}$$

where $\mathbb{E}[\cdot]$ is the expectation over the time-varying system parameters, such as users' content requests and the association between end devices and BSs. The constraint in (7b) guarantees the limits of bandwidth and cache capacity. The constraints in (7c) and (7d) guarantee that at most one content is replaced and each content request can only be met by one place, respectively. This optimization problem includes the integer variables $\mathcal{A}_b^t, \boldsymbol{z}_b^t$ and the continuous variables $\boldsymbol{w}_b^t$. According to (1), (2), and the edge-end latency $l_{b,m,f}^E = \kappa_f / v_{b,m}^t$, the continuous variables $\boldsymbol{w}_b^t$ are located in the denominator. Hence, this problem is a mix-integer nonlinear programming problem (MINLP), which is NP-hard [42]. The conventional methods are difficult to solve it.

IV. Decentralized Recommendation-Enabled Edge Caching Framework

In this section, we first analyze and decompose the formulated problem into two subproblems. Then, we present a decentralized recommendation-enabled edge caching framework to solve this optimization problem. Finally, we give a detailed analysis of computational complexity.

A. Problem Analysis and Decomposition

To address the formulated problem, we first analyze the recommendation-enabled edge caching system at a certain BS in one time slot, as shown in Fig. 3. At the beginning of one time slot, several users served by the BS initiate content requests. We need to first determine the request processing action to deliver contents to users. Meanwhile, if the requested contents are not cached at the edge server, we should determine whether to recommend similar contents to users, i.e., soft hits. We then optimize the allocation of wireless resource to reduce the transmission latency between the BS and users. Finally, the edge server should replace the cached contents to hit more content requests, i.e., the cache replacement action. Towards this end, we need to determine the local user requests and the cache replacement. Therefore, the recommendation-enabled edge caching process is divided into the local request processing and the cache replacement processing, where the local request processing includes the content recommendation and the resource allocation. This complex optimization problem can be decomposed into two subproblems: *local request processing* and *cache replacement processing*.

We should further discuss the relationship of the local request processing and the cache replacement processing. We

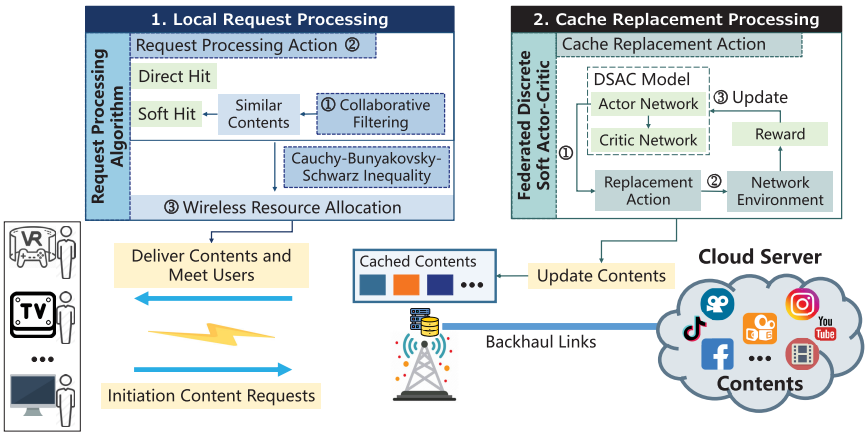


Fig. 3. Proposed decentralized recommendation-enabled edge caching framework in the considered network.

rewrite the reward in (6) as $\mathcal{R}_b^t(\mathcal{S}_b^t, \mathcal{A}_b^t) = \lambda_1\mu_b^t + \lambda_2\nu_b^t + \lambda_3\mathcal{L}_b^t + \lambda_3 \sum_{f\in\{0\}\cup\mathcal{F}} a_{b,f}^t l_f^C$. We find that the request processing action and the cache replacement action are independent and can be determined separately in one time slot. Therefore, two subproblems can be solved independently. For the local request processing, we reduce the problem in (7) for BS $b \in \mathcal{B}$ in time slot $t \in \mathcal{T}$ from maximization to minimization as

$$\min_{\boldsymbol{z}_b^t, \boldsymbol{w}_b^t} (\lambda_1\mu_b^t + \lambda_2\nu_b^t + \lambda_3\mathcal{L}_b^t), \text{ s.t. (7b), (7d)-(7f),} \tag{8}$$

which includes the integer variables $\boldsymbol{z}_b^t$ and the continuous variables $\boldsymbol{w}_b^t$. According to the above analysis, this optimization problem is also MINLP and NP-hard. For the cache replacement processing, each policy of the BS should be optimized iteratively to improve the user QoE.

B. Local Request Processing

1) Similarity Matrix Building: We present a content recommendation algorithm based on collaborative filtering (CF) to build a similarity matrix $\Phi_{F\times F}^m$ for user $m \in \mathcal{M}$. The idea of CF is to recommend contents to users based on their previous content preferences and the choices of other users with similar interests. CF first calculates the similarity between contents f and j based on correlation, calculated by

$$Sim(f,j) = \frac{|N(f) \cap N(j)|}{\sqrt{|N(f)||N(j)|}}, \quad \forall f, j \in \mathcal{F}, \tag{9}$$

where $|N(f)|$ and $|N(j)|$ are numbers of users who like contents f and j, respectively; $|N(f) \cap N(j)|$ is the number of users who like contents f and j at the same time. We can infer that, in CF, two contents have similarity because they are liked by many users with similar interests.

Different users perform different behavior habits and content performances, which will affect the result accuracy of the CF algorithm. For instance, a user with high activity may like more contents in the catalog, which will overestimate the similarity of the user with low activity. Current similarities cannot meet the personalization. Therefore, we need to combine the rating γ_m^f of content f by user m and the similarity in (9). Then the similarity score of contents f and j can be calculated by $\phi_{f,j}^m = \gamma_m^f Sim(f,j), \forall m \in \mathcal{M}, \forall f, j \in \mathcal{F}$.

2) Local Request Processing Algorithm: Edge server $b \in \mathcal{B}$ receives a set of content requests $\{\rho_{m,f}^t\}_{m\in\mathcal{M}_b^t, f\in\mathcal{F}}$ at the beginning of time slot $t \in \mathcal{T}$. This edge server should determine the optimal $\boldsymbol{z}_b^{t*}$ and $\boldsymbol{w}_b^{t*}$ to minimize the system cost. Note that the resource allocation $\boldsymbol{w}_b^t$ make sense and can be determined, only after $\boldsymbol{z}_b^t$ are determined in the problem (8). Thus, $\boldsymbol{z}_b^t$ and $\boldsymbol{w}_b^t$ are determined sequentially.

As the number of users served by each edge server is finite, we can utilize a greedy algorithm to find the optimal request processing action, as shown in Algorithm 1. Each edge server $b \in \mathcal{B}$ should determine all requests of users $\mathcal{M}_b^t$. For $\rho_{m,f}^t = 1$, if f has been cached at edge server b, we should choose direct hits $z_m^{E,t} = 1$. When the cached contents at edge server b do not include content f, we consider recommending a similar content to user m, i.e., soft hit $z_m^{S,t} = 1$. We need to get the similarity matrix $\Phi_{F\times F}^m$ for user m and then get top i similar contents (denoted as $\mathcal{F}_{sim}^t$) about content f. If one of the similar contents appears in the cached contents, we choose the soft hit, otherwise, the direct hit from the cloud server is chosen, i.e., $z_m^{C,t} = 1$. When the integer variables $\boldsymbol{z}_b^t$ are determined, we can obtain a closed form of optimizing the continuous variables $\boldsymbol{w}_b^t$ according to Lemma 2.

Lemma 2: Once the integer variables $\boldsymbol{z}_b^{t*}$ have been determined, the optimal solution of $\boldsymbol{w}_b^{t*}$ can be calculated in a closed form as

$$w_{b,m}^{t*} = \frac{W_b \sqrt{\sum_{f\in\mathcal{F}} \frac{\rho_{m,f}^t((1-z_m^{S,t})\kappa_f + z_m^{S,t}\kappa_j)}{\log_2(1+\frac{p_b|h_{b,m}|^2}{\sigma^2})}}}{\sum_{m\in\mathcal{M}_b^t} \sqrt{\sum_{f\in\mathcal{F}} \frac{\rho_{m,f}^t((1-z_m^{S,t})\kappa_f + z_m^{S,t}\kappa_j)}{\log_2(1+\frac{p_b|h_{b,m}|^2}{\sigma^2})}}}, \quad \forall m \in \mathcal{M}_b^t, \forall b \in \mathcal{B}, \forall t \in \mathcal{T}. \tag{10}$$

Proof: Please see Appendix B. ■

C. Cache Replacement Processing

We propose a FDSAC algorithm to cope with the cache replacement problem. Specifically, we first propose a multi-agent discrete variant of SAC, which consists of an actor network and a critic network. Then, we improve the actor network by integrating LSTM to capture the time-series

Algorithm 1 Request Processing Algorithm at BS b

Input: $\mathcal{M}_b^t, \boldsymbol{s}_b^t, i$, and content information.
Output: $\boldsymbol{z}_b^{t*}$ and $\boldsymbol{w}_b^{t*}$.
Initialization $\boldsymbol{z}_b^{t*} = \mathbf{0}$ and $\boldsymbol{w}_b^{t*} = \mathbf{0}$;
for $m \in \mathcal{M}_b^t$ **do**
  Find f when $\rho_{m,f}^t = 1$;
  if $s_{b,f}^t = 1$ **then**
    Set $z_m^{E,t} = 1$;
  else
    Build $\Phi_{F\times F}^m$ by a content recommendation algorithm;
    Get top i similar contents $\mathcal{F}_{sim}^t$ according to $\Phi_{F\times F}^m$;
    if $\exists i \in \mathcal{F}_{sim}^t,\ s.t.\ s_{b,i}^t = 1$ **then** Set $z_m^{S,t} = 1$;
    else Set $z_m^{C,t} = 1$;
  end
  Get w_m^{t*} based on (10);
end

dependence of replacement action. Finally, we improve the critic network by integrating FL with the attention mechanism to protect user privacy.

1) Proposed Discrete Soft Actor-Critic: SAC is an advanced DRL algorithm with a maximum entropy objective. By integrating the maximum entropy objective to the reward, SAC has equal probabilities to explore unknown state space, i.e., all actions will be effectively explored and exploited [43]. The actor network is responsible for learning the policy to make the replacement action. The critic network is responsible for evaluating the policy to enhance training efficiency. To overcome the limitation of SAC for continuous actions, we transform the policy output from a density $\pi(\boldsymbol{a}^t|\mathcal{S}^t)$ to a probability $\pi(\mathcal{S}^t)$ for the discrete scenario, named as DSAC. Thus, the policy π of DSAC can maximize both the reward and the entropy, i.e.,

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{i=t}^{\infty} \gamma^{i-t}\left(r^i + \delta\mathcal{H}(\pi(\mathcal{S}^i))\right)\right], \quad (11)$$

where $\mathcal{H}(\pi(\mathcal{S}^t)) = \mathbb{E}[-\log\pi(\mathcal{S}^t)]$ is the entropy which can measure the uncertainty of random variables; δ is the temperature parameter to balance the relative importance of the entropy term versus the reward.

To evaluate all finite discrete actions, the soft Q-function should output the Q-value of each possible action, i.e., from $Q: \mathcal{S}^t \times \boldsymbol{a}^t \to \mathbb{R}^{2|\boldsymbol{a}^t|}$ to $Q: \mathcal{S}^t \to \mathbb{R}^{|\boldsymbol{a}^t|}$. Accordingly, the soft Q-function is redefined as

$$Q(\mathcal{S}^t) = r^t + \gamma\mathbb{E}\left[V(\mathcal{S}^{t+1})\right], \quad (12)$$

where $V(\mathcal{S}^t) = \pi(\mathcal{S}^t)^T\left[Q(\mathcal{S}^t) - \delta\log\pi(\mathcal{S}^t)\right]$ represents the soft value function with the entropy-augmented accumulated return.

Remark 1: For a fixed policy π, its soft Q-value can be calculated iteratively by the soft Bellman backup operator τ^π, i.e., $Q^{t+1} = \tau^\pi Q^t$.

During the policy updating, we have $\pi_{new}(\mathcal{S}^t) \propto \exp(Q^{\pi_{old}}(\mathcal{S}^t))$, i.e., two variables are directly proportional. Different from conventional off-policy methods by maximizing the Q-value, the policy updating of DSAC is proportional to the exponential distribution of $Q(\mathcal{S}^t)$. But in actual operation, we still output the policy as a Gaussian distribution to facilitate the policy processing, and minimize the gap between two distributions by the Kullback-Leibler (KL) divergence, i.e.,

$$\pi_{new} = \arg\min_{\pi'} D_{\mathrm{KL}}(\pi'(\mathcal{S}^t) \parallel \frac{\exp\left(1/\delta Q^{\pi_{\text{old}}}\left(\mathcal{S}^t\right)\right)}{Z^{\pi_{\text{old}}}\left(\mathcal{S}^t\right)}), \quad (13)$$

where $D_{\mathrm{KL}}(\cdot)$ is the KL divergence; $Z^{\pi_{\text{old}}}\left(\mathcal{S}^t\right)$ is used to normalize the distribution of Q-value.

Lemma 3: Minimizing the KL divergence between the policy distribution and the exponential distribution of Q-function is equivalent to maximize our objective function. The new policy in (13) can lead to $Q^{\pi_{\text{new}}}\left(\mathcal{S}^t\right) \geq Q^{\pi_{\text{old}}}\left(\mathcal{S}^t\right)$.

Proof: Please see Appendix C. ∎

Similar to the policy iterative solution of canonical RL, two processes of soft policy evaluation (12) and soft policy improvement (13) can be solved iteratively to find the optimal policy finally. This scheme ensures that the optimal policy is discrete in the state-action space and can be found within finite iterations. In this paper, we use function approximates (i.e., deep neural networks (DNNs)) to find the optimal policy of DSAC. We denote $Q_\theta(\mathcal{S}^t)$ and $\pi_\vartheta(\mathcal{S}^t)$ as the parameterized soft Q-function and the parameterized policy function by DNNs, respectively, where θ and ϑ are parameters of DNNs. We minimize the mean-square error of the soft Bellman residual as the objective function of $Q_\theta(\mathcal{S}^t)$, i.e.,

$$J_Q(\theta) = \mathbb{E}_{\mathcal{S}^t\sim\mathcal{D}}[\frac{1}{2}(Q_\theta\left(\mathcal{S}^t\right) - \hat{Q}\left(\mathcal{S}^t\right))^2], \quad (14)$$

where $\mathcal{D}$ is a replay buffer, which stores the historical transition (i.e., $\boldsymbol{d}^t = \{\mathcal{S}^t, \boldsymbol{a}^t, r^t, \mathcal{S}^{t+1}\} \in \mathcal{D}$); $\hat{Q}\left(\mathcal{S}^t\right)$ is the target soft Q-function. We can further obtain the gradient of $J_Q(\theta)$ as follows

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla Q_\theta(\mathcal{S}^t)(Q_\theta(\mathcal{S}^t) - (r^t + \gamma V(\mathcal{S}^{t+1}))). \quad (15)$$

Similarly, the objective function of $\pi_\vartheta(\mathcal{S}^t)$ is to minimize the KL divergence

$$J_\pi(\vartheta) = \mathbb{E}_{\mathcal{S}^t\sim\mathcal{D}}\left[D_{\mathrm{KL}}\left(\pi_\vartheta(\mathcal{S}^t) \parallel \frac{\exp\left(1/\delta Q_\theta\left(\mathcal{S}^t\right)\right)}{Z_\theta\left(\mathcal{S}^t\right)}\right)\right]. \quad (16)$$

As $Z_\theta(\cdot)$ is a partition function and is independent of ϑ, we can omit it. Additionally, the policy outputs the exact action distribution, the re-parameterization technique is not used like SAC with the continuous action. Thus, we have Remark 2.

Remark 2: The objective function in (16) can be calculated directly as

$$J_\pi(\vartheta) = \mathbb{E}_{\mathcal{S}^t\sim\mathcal{D}}[\pi_\vartheta(\mathcal{S}^t)^T(\delta\log\pi_\vartheta(\mathcal{S}^t) - Q_\theta(\mathcal{S}^t))]. \quad (17)$$

The gradient of (17) is expressed as

$$\begin{aligned}\hat{\nabla}J_\pi(\vartheta) = {} & \nabla_\vartheta\pi_\vartheta(\mathcal{S}^t)^T(\delta\log\pi_\vartheta(\mathcal{S}^t) - Q_\theta(\mathcal{S}^t)) \\ & + \pi_\vartheta(\mathcal{S}^t)^T\delta\nabla_\vartheta\log\pi_\vartheta(\mathcal{S}^t). \end{aligned} \quad (18)$$

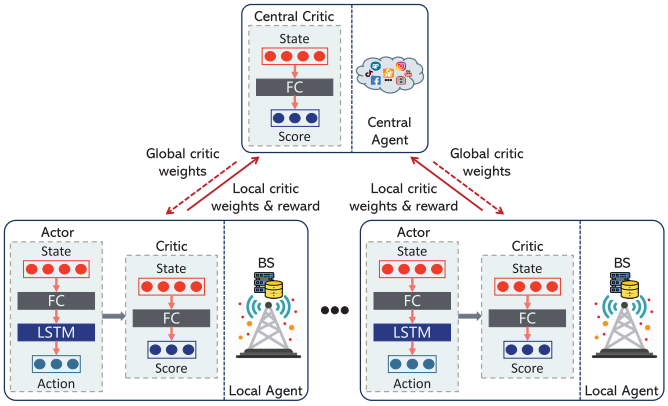


Fig. 4. Diagram of FDSAC implementation.

2) Neural Network Design: We consider that the actor network and the critic network of DSAC are both constructed by fully-connected networks. However, pure fully-connected networks cannot obtain fully the hidden content request patterns. Additionally, the state with historical requests will increase the input space, making DSAC difficult to learn the optimal policy. Motivated by LSTM with memory mechanism, we utilize LSTM to improve the actor network to capture the hidden content request patterns, shown in Fig. 4. LSTM is suitable for hidden information extraction and has been widely used to learn the temporal dependence of sequential states and predict the future variation of time series. Specifically, we add a LSTM layer between the hidden layer and the output layer, which includes a set of hidden neurons. The output of the fully-connected layer and a hidden matrix $H(t)$ are combined as an input, where the hidden matrix is responsible for recording the full history information of observation and actions dynamically.[4] Therefore, the LSTM layer can remember the feature representations of past time slots for the replacement action at the current time slot.

3) Algorithm Implementation: In the proposed framework, there are B DSAC models deployed at B edge servers and a critic network of DSAC deployed at the cloud server, as shown in Fig. 4. Different from existing FL aggregating all local models to the global model, we separate the actor network and the critic network of DSAC and only aggregate the parameters of all local critic networks to the central critic network. This is because, in theory, all critic networks should have the same evaluation criteria for the current network environment, but the actor network should be different due to the request heterogeneity of users. We further utilize a attention mechanism to strengthen the better agent and decrease the worse agent based on all rewards of agents, expressed as

$$\vartheta_{global} = \frac{\sum_{b\in\mathcal{B}} r_b^{\text{total}} \times \vartheta_b}{\sum_{b\in\mathcal{B}} r_b^{\text{total}}}, \tag{19}$$

where $r_b^{\text{total}} = \sum_{t\in\mathcal{T}} r_b^t$ is the total reward of BS b within T time slots; ϑ_{global} are the parameters of the central critic network; ϑ_b are the parameters of the critic network at BS b.

[4]Each hidden matrix is also stored in a replay buffer in each time slot.

Algorithm 2 FDSAC Algorithm

1 Initialization $\mathcal{B}, \mathcal{M}, \mathcal{F}, \mathcal{T}$, batch size N, and aggregation Δ;
2 Initialization $\{\mathcal{D}_b, \theta_b, \vartheta_b\}_{b\in\mathcal{B}}, \vartheta_{global}$, and LSTM;
3 **while** *episode=1,2,...* **do**
4 Set $r_b^{\text{total}} = 0$ for all $b \in \mathcal{B}$;
5 **for** $t \in \mathcal{T}$ **do**
6 **for** $b \in \mathcal{B}$ **do**
7 Get replacement action $\boldsymbol{a}_b^t \leftarrow \pi_{\theta_b}(\cdot|\mathcal{S}_b^t)$;
8 Get $\boldsymbol{z}_b^t$ and $\boldsymbol{w}_b^t$ by Algorithm 1;
9 Get reward r_b^t and next state $\mathcal{S}_b^{t+1}$;
10 r_b^{total} += r_b^t;
11 Store a tuple $\boldsymbol{d}_b^t$ into $\mathcal{D}_b$;
12 Randomly draw a batch of N transitions as $\mathcal{D}_b^N \in \mathcal{D}_b$;
13 Update θ_b and ϑ_b by (15) and (18), respectively;
14 **end**
15 **end**
16 **if** *episode mod* $\Delta = 0$ **then**
17 Upload all $\{r_b^{\text{total}}, \vartheta_b\}_{b\in\mathcal{B}}$ to update ϑ_{global} by (19);
18 Distribute new ϑ_{global} to update all $\{\vartheta_b \leftarrow \vartheta_{global}\}_{b\in\mathcal{B}}$;
19 **end**
20 **end**

This aggregating manner of FL has many advantages: i) training and inference are performed locally, which reduces data transmission and the risk of privacy leakage; ii) different local models with the unique actor network can maintain individualization; iii) the attention-empowered FL algorithm can accelerate training.

4) Algorithm Training: In the proposed FDSAC Algorithm, each edge server is deployed with an agent (i.e., local DSAC model), which trains the local model independently and aggregates the model parameters cooperatively. As shown in Algorithm 2, the local agent in BS b needs to make the request processing action and the cache replacement action locally, steps 4-15. First, the agent observes its local state $\mathcal{S}_b^t$ and generates its replacement action $\boldsymbol{a}_b^t$. The optimal $\boldsymbol{z}_b^{t*}$ and $\boldsymbol{w}_b^{t*}$ are calculated by Algorithm 1. Therefore, we can further obtain the current reward r_b^t and the next state $\mathcal{S}_b^{t+1}$ and store the tuple in a replay buffer. Then, steps 11-13 update and learn the actor network and the critic network based on a batch $\mathcal{D}_b^N$. After many iterations, when the episode equals a multiple of the number of aggregation, all agents upload their parameters of critic networks and rewards $\{r_b^{\text{total}}, \vartheta_b\}_{b\in\mathcal{B}}$ to the central critic network, steps 16-19. The cloud server calculates the central critic parameters ϑ_{global} by (19). Finally, the central critic parameters are distributed to update all local critic networks.

D. Computational Complexity

Each BS $b \in \mathcal{B}$ needs to train and infer the neural network in each time slot. First, according to Algorithm 1,

TABLE II
PARAMETER SETTINGS

	Parameter	Value		Parameter	Value
Network Environment	$W_b, b \in \mathcal{B}$	20 MHz	FDSAC	Number of neurons	128
	$\kappa_f, f \in \mathcal{F}$	[5, 10] Mbit		Learning rate	0.0003
	$p_b, b \in \mathcal{B}$	30 dBm		Optimizer	Adam
	σ^2	-100 dBm		N	256
	v_c	50 Mbps		γ	0.99
	α	1		δ	0.2
	T	50		Δ	50
	$K_b, b \in \mathcal{B}$	100 Mbit		$\lvert\mathcal{D}\rvert$	10000

we can determine the computational complexity of the request processing algorithm at BS b as $O(|\mathcal{M}_b^t|)$. To determine the computational complexity of Algorithm 2, we denote Ψ as the number of multiplication operations in the neural network. The computational complexity of backpropagation for the training of one transition is $O(\Psi)$. As the local DSAC model trains parallel, we only consider the computational complexity of one local model. Thus, we determine the computational complexity of DSAC at BS b as $O(\Psi ENT|\mathcal{M}_b^t|)$, where E is the number of episodes; N is the number of transitions sampled in each round of training.

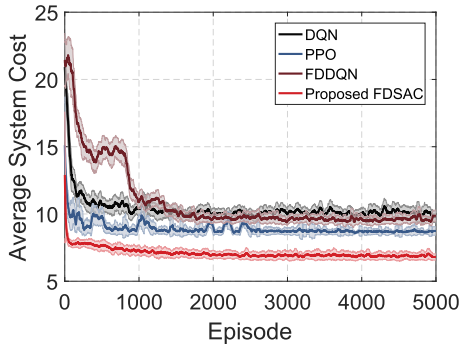
V. TRACE-DRIVEN PERFORMANCE EVALUATION

A. Setup and Baselines

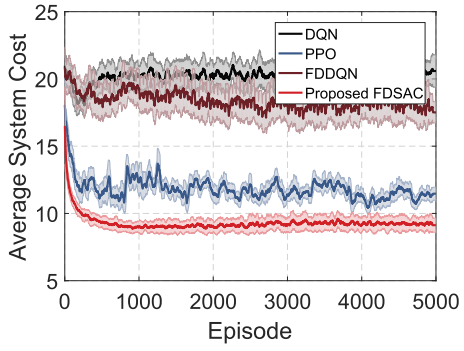
We consider a scenario with 40 end devices and four edge servers, each of which covers a circular area with radius of 1 km. We assume that the association is known in each time slot. As the latency affects the user QoE more, $\lambda_1, \lambda_2, \lambda_3$ are set as 1/5, 1/5, and 3/5. Unless otherwise stated, the default parameter settings about network environments and FDSAC are given in Table II. Additionally, the channel gain is modeled as $h_{b,m} = 30.6 + 36.7 \log_{10} d$ dB. The actor network consists of an input layer, an output layer, two hidden layers, and an LSTM layer. The architecture of the critic network is the same as the actor network except for no LSTM.

To make the experiments more practical, we use both synthetic and real datasets [44]. i) Synthetic dataset: we consider the scenario that 100 users request the catalog with 200 contents. Each content size is set as [5], [10] Mbit. The end devices and the catalog are populated with values drawn from random probability distributions. ii) MovieLens dataset: this dataset includes the ratings of movies and has been widely used in recommender systems. We utilize different samples of 600 users and 8,000 contents of the catalog. For convenience, we assume that each content size is also set as [5], [10] Mbit. In both synthetic and MovieLens datasets, the request sequence of each user follows the Zipf distribution.

To evaluate the performances of the proposed algorithms, we consider the following five caching algorithms for comparison. i) LRU [12]: The LRU content will be replaced first. ii) LFU [13]: The LFU contents will be replaced first. iii) First in First Out (FIFO) [45]: The oldest contents will be replaced first. iv) DQN [36]: DQN is a classic RL, which can obtain near-optimal results in small-scale settings. v) Federated Discrete DQN (FDDQN): Each edge server is deployed with a DQN model, where a center aggregator can federate all DQN models. vi) Proximal Policy Optimization (PPO) [46]: PPO is a DRL algorithm with deterministic policy, which has been widely used in existing studies.



(a)



(b)

Fig. 5. Convergence of four RL-based caching algorithms for: (a) Synthetic dataset; (b) MovieLens dataset.

The experiments are conducted on a 64-bit Windows 10 desktop, which is equipped with a 32G RAM, an Intel i7 CPU, and a GTX1050-Ti GPU. The programming tool is "Python 3.6" with deep learning library "PyTorch 1.11".

B. Convergence Performance

In this subsection, we evaluate the convergence performances of four RL-based caching algorithms under different neural network hyperparameters. Specifically, we use the average system cost as the performance criterion (i.e., average (6) within one episode). The solid curves and the shaded areas represent the mean and standard deviation of the average system cost, respectively. The x-axis and y-axis are episode and the average system cost, respectively. We consider 5,000 episodes, where each episode includes 50 time slots.

Fig. 5 shows the convergence curve of FDSAC, DQN, FDDQN, and PPO for synthetic and MovieLens datasets with the default setting. Clearly, as the episode increases, the proposed FDSAC algorithm (red line) converges a lower value of average system cost for both two datasets compared with DQN and PPO. This is because FDSAC with the maximum entropy objective has the ability to explore more action spaces. Besides, LSTM can accelerate the training of actor network. Particularly, for the synthetic dataset, we can see that FDSAC decreases drastically in the first 150 episodes and converges

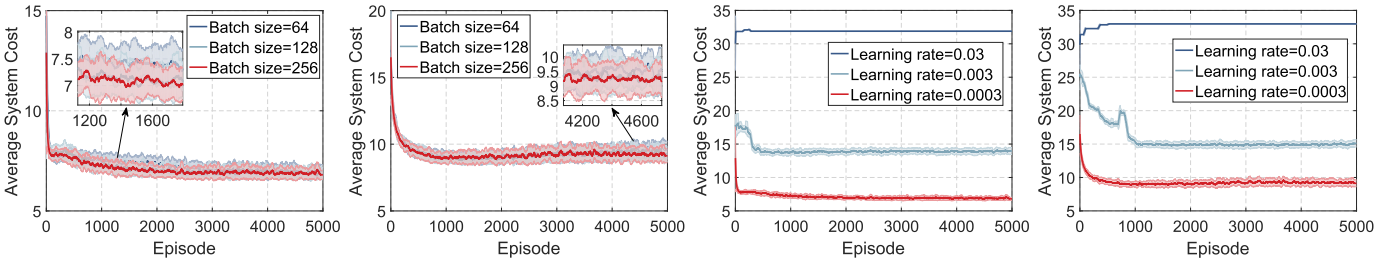


Fig. 6. Convergence of FDSAC under different batch sizes and learning rates: (a) Synthetic dataset with different batch sizes; (b) MovieLens dataset with different batch sizes; (c) Synthetic dataset with different learning rates; (d) MovieLens dataset with different learning rates.

to about 6.5 finally. As PPO integrates actor-critic structure, PPO is better than DQN. However, PPO and DQN are more fluctuant and unstable, which are more obvious for the MovieLens dataset. FDDQN is better than DQN, which can converge lower value of average system cost. For the MovieLens dataset, the proposed FDSAC algorithm decreases slowly compared with the synthetic dataset. Although DQN does not work well because these algorithms are more difficult to learn from the real complex scenario, FDDQN is stable.

It is very important to investigate the convergence of deep learning algorithms under different neural network hyperparameters. Fig. 6 shows the convergence curve of FDSAC under different batch sizes and learning rates. The batch size represents the number of experience transitions sampled in each training round. The step size represents moving towards the minimum of the loss function in each training round. From Fig. 6(a), as the batch size increases from 64 to 256, the convergence speed increases for the synthetic dataset. Meanwhile, the standard deviation (i.e., shaded area) is smaller. The proposed FDSAC algorithm with the batch size $= 256$ has a lower value of average system cost from Fig. 6(b). From Fig. 6(c) and Fig. 6(d), we can see that the learning rate $= 0.0003$ leads to a relatively fast convergence and small converged value. As the learning rate increases from 0.0003 to 0.03, the learning efficiency of FDSAC deteriorates, FDSAC especially cannot converge at all. This is because an excessive learning rate will cause the model to quickly oscillate beyond the effective range. Meanwhile, Figs. 6(c) and 6(d) show different trends about the system cost because the MovieLens dataset is a real dataset with complex user behavior habits and content performances. Thus, we choose the batch size $= 256$ and the learning rate $=$ 0.0003 as default settings.

To verify the effectiveness of LSTM, we illustrate the effects of LSTM on the proposed algorithm for two datasets in Fig. 7. For two datasets, the proposed FDSAC (red line) performs better than FDSAC without LSTM. FDSAC with LSTM converges faster and lower because LSTM can capture the hidden information of historical observations and actions. Specifically, compared with Fig. 7(a) and Fig. 7(b), LSTM has greater effects about convergence for the MovieLens dataset. This is because the real dataset has more complex and dynamic information about users' requests and content performances than the synthetic dataset. The proposed FDSAC with LSTM is easy to capture the hidden information to learn a better policy about replacement action.

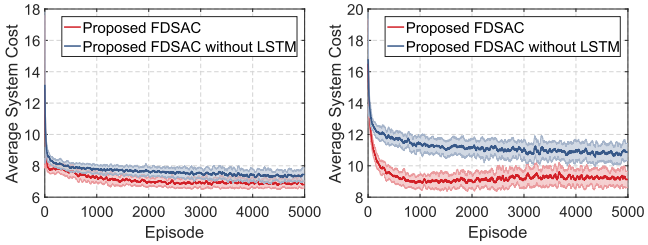


Fig. 7. Effects of LSTM on the proposed algorithm for: (a) Synthetic dataset; (b) MovieLens dataset.

C. Effects of Different BS Indexes

Fig. 8 and Fig. 9 show the effects of different BS indexes for the synthetic dataset and the MovieLens dataset, respectively. In Fig. 8, we vary the BS index from 1 to 4. Generally, we can see that four RL-based caching algorithms have better performance in terms of the average system cost compared with three rule-based caching algorithms under all cases with different BS indexes from Fig. 8(a). This is because four RL-based algorithms have the learning ability to generate a more judicious cache replacement action. Particularly, FDSAC (red bar) performs the best with the lowest average system cost. As the BS index increases, the average system costs of seven caching algorithms also increase because the proposed system needs more costs to deal with more caching requests. Especially, Fig. 8(c) illustrates that four RL-based caching algorithms have more average similarity cost compared with three rule-based caching algorithms. This is because agents (i.e., RL-based models) choose to retrieve the similar contents cached at the edge servers to users due to soft hits, which can greatly reduce the transmission latency. Although soft hits will cause more similarity cost, the similarity cost is dwarfed compared with the latency cost. Besides, the cache hit cost can be greatly decreased in Fig. 8(d).

In Fig. 9, performances of seven caching algorithms for the MovieLens dataset are similar to the synthetic dataset. It is worth noting that DQN performs poorly. At the BS index $= 4$, the difference of average system cost between DQN and FDSAC increases from 1 to 10 for the synthetic dataset and the MovieLens dataset, respectively. The growth rate is as high as 900%. As DQN with a simple structure and policy, it cannot capture and learn the feature of the complex real dataset. FDDQN is better than DQN. In general, FDSAC

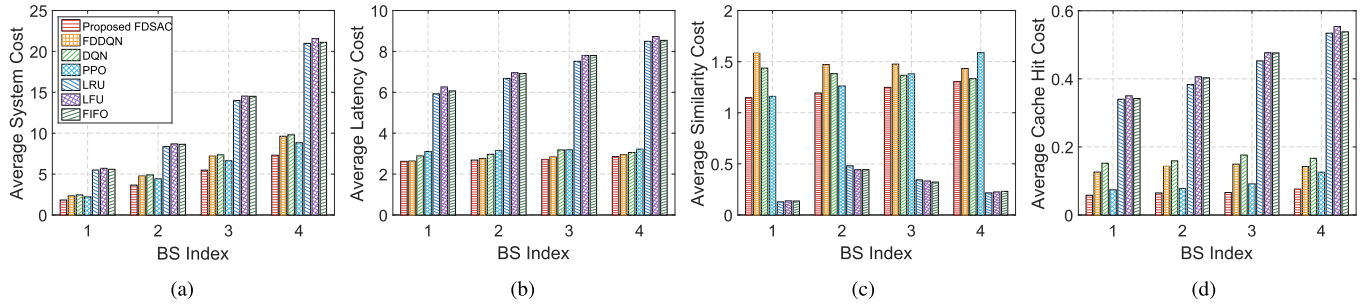


Fig. 8. Effects of different BS Indexes for Synthetic dataset: (a) average system cost; (b) average latency cost; (c) average similarity cost; (d) average cache hit cost.

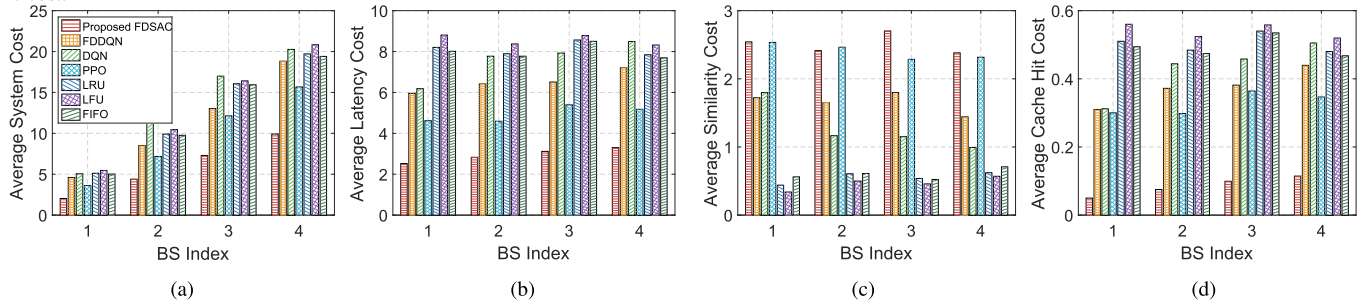


Fig. 9. Effects of different BS Indexes for MovieLens dataset: (a) average system cost; (b) average latency cost; (c) average similarity cost; (d) average cache hit cost.

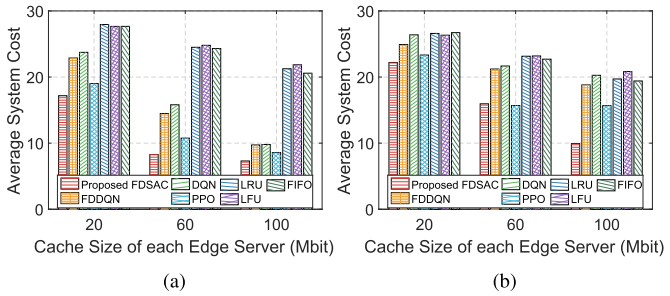


Fig. 10. Effects of different cache sizes of edge servers for: (a) Synthetic dataset; (b) MovieLens dataset.

has more stability and better performance in terms of the average system cost for both the synthetic dataset and the MovieLens dataset.

D. Effects of Different Cache Sizes of Edge Servers

Fig. 10 compares the performance of the proposed FDSAC algorithm with the baselines in terms of the average system cost versus different cache sizes of each edge server. For both two datasets, FDSAC always outperforms the other six algorithms on the average system cost at different cache sizes. Meanwhile, from Figs. 10(a) and 10(b), as the cache size of each edge server increases, the average system cost of each algorithm decreases because the edge servers with larger cache capacity can cache more contents for users. Particularly, FDSAC has a faster descent process compared with the other three RL-based algorithms. There is an interesting result that the average system cost of FDSAC does not decrease when the cache size of each edge server increases to 60 Mbit for the Synthetic dataset in Fig. 10(a). This may be because the cache size with 60 Mbit is sufficient for FDSAC to make the optimal caching strategy. The bigger cache size has a small contribution to reducing the cost. But for the MovieLens dataset in Fig. 10(b), the bigger cache size of each edge server can help all algorithms reduce the average system cost.

E. Effects of Different Numbers of End Devices

Moreover, we investigate the effects of different numbers of end devices for two datasets in Fig. 11 and Fig. 12. We vary the number of end devices from 20 to 60. In Fig. 11(a), FDSAC achieves a lower average system cost than the other caching algorithms, especially when the number of end devices is large. This is because FDSAC can efficiently cope with the dynamic scenario with large-scale end devices. Besides, the aid of LSTM can make FDSAC capture the hidden information to make more proper actions. In Fig. 11(a), when the number of end devices increases to 40, the average system costs of FDDQN, DQN, and PPO relatively accelerate growth. The slope increases from about 0.35 to 0.65. In Fig. 11(b), during numbers of end devices in $[20, 36]$, the average latency costs of FDSAC, FDDQN, DQN, and PPO are almost the same. But DQN and PPO perform worse in the average similarity cost and the average cache hit cost from Fig. 11(c) and Fig. 11(d). This may be because FDSAC can capture user preferences at the same time. Although three rule-based algorithms have lower average similarity costs, they have larger average latency costs and average cache hit costs. In Fig. 12, performances of seven caching algorithms for the MovieLens dataset are similar to the synthetic dataset, except for FDDQN and DQN.

F. Effects of Soft Hits

Finally, we illustrate the effects of soft hits on seven caching algorithms for two datasets in Fig. 13. In general, we can see that seven caching algorithms with soft hits achieve lower average system costs. This indicates that the proposed recommendation-enabled edge caching framework can dramatically reduce the system cost and improve the user QoE. For FDSAC, the difference (blue line) between two average system costs of soft hits and direct hits is about 5.25 and 14 for the two datasets. This is because the real

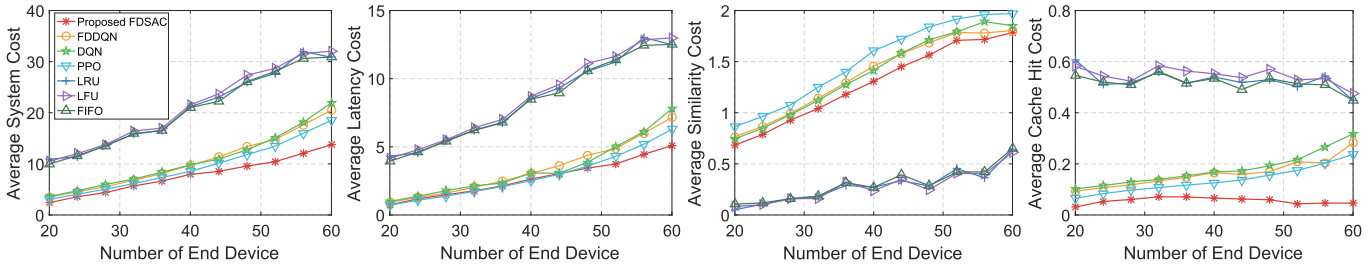


Fig. 11. Effects of different numbers of end devices for Synthetic dataset: (a) average system cost; (b) average latency cost; (c) average similarity cost; (d) average cache hit cost.

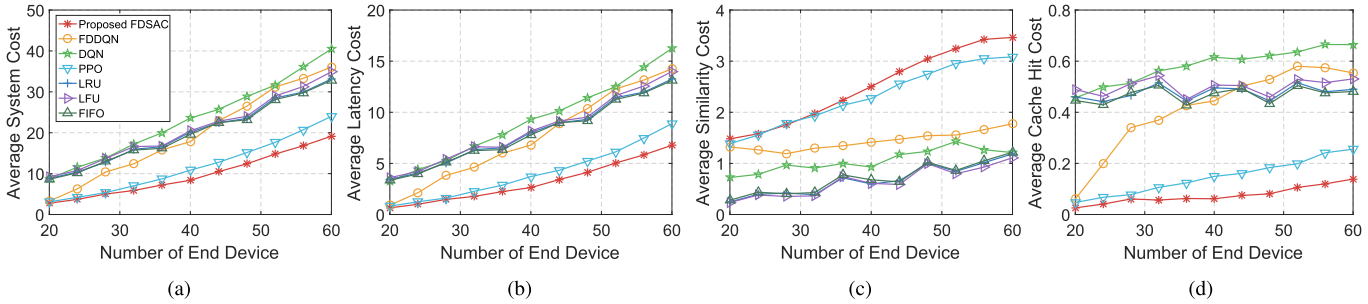
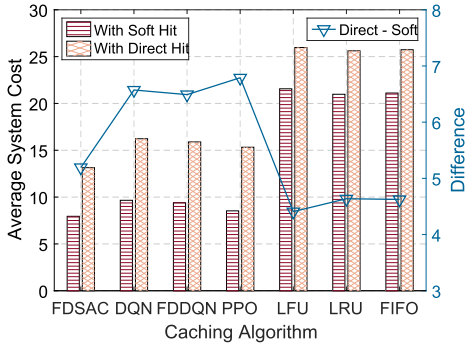


Fig. 12. Effects of different numbers of end devices for MovieLens dataset: (a) average system cost; (b) average latency cost; (c) average similarity cost; (d) average cache hit cost.



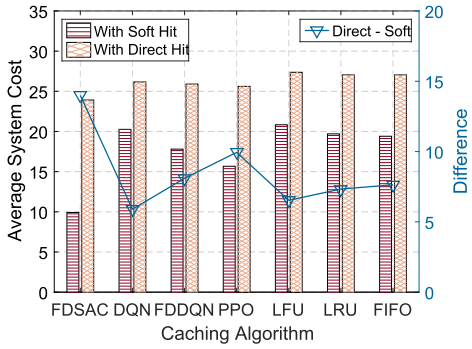


Fig. 13. Soft hits versus direct hits in terms of average system cost: (a) Synthetic dataset; (b) MovieLens dataset.

user preference is more complex and similar contents of users are more scattered. In contrast, three rule-based caching algorithms are less affected by the manners of soft hits and direct hits. The difference between two average system costs of direct hits and soft hits is about 4.6 in Fig. 13(a). This is because they can not learn content preferences of users.

VI. Conclusion

In this paper, we have proposed a decentralized recommendation-enabled edge caching framework in mobile edge-cloud computing (a case of two-tier computing) networks. Particularly, we have focused on minimizing the system cost (reflecting the user QoE) under certain constraints. We have further proposed a novel FDSAC algorithm to address the formulated optimization problem. To implement the experiments more practically, we have used both synthetic and MovieLens datasets. Trace-driven simulation results have demonstrated that FDSAC significantly outperforms three existing RL-based algorithms (DQN, FDDQN, and PPO) and three rule-based algorithms (LRU, LFU, and FIFO) in terms of algorithm convergence and average system cost. Besides, the average system cost of recommendation-enabled edge caching is 2.4 times less on the real dataset than that of conventional edge caching.

Appendix

A. Proof of Lemma 1

According to Fact 1, two different contents have a similarity for a user. This means that a user has a similar preference for two content, i.e., like or dislike at the same time. When one of the two similar contents is not easy to retrieve but the other is easy, the user is more willing to choose the easily accessible content. Besides, the greater the similarity between the two, the more willing the user is to accept the other as a candidate. We utilize the similarity matrix to represent the probability of reception. For example, $\phi^m_{f,j} = 1, \forall f, j \in \mathcal{F}, \forall m \in \mathcal{M}$ represents that two contents are completely different, user m does not accept to replace content f with content j; $\phi^m_{f,f} = 1, \forall f \in \mathcal{F}, \forall m \in \mathcal{M}$ represents that two contents are identical and completely similar. Further, direct hits and soft

hits in Definition 1 can strengthen the edge caching system to improve the user QoE. This completes the proof.

B. Proof of Lemma 2

Once $\boldsymbol{z}_b^t, \forall b \in \mathcal{B}, \forall t \in \mathcal{T}$ have been determined, the problem in (8) is only related to $\boldsymbol{w}_b^{t*} = (w_{b,m}^t)_{m\in\mathcal{M}_b^t}$ and can be transformed into the following

$$\begin{aligned}\boldsymbol{w}_b^{t*} = \arg\min_{\boldsymbol{w}_b^t} \lambda_3 \sum_{m\in\mathcal{M}_b^t}\sum_{f\in\mathcal{F}} \rho_{m,f}^t \\ ((1-z_m^{S,t})l_{b,m,f}^E + z_m^{S,t} l_{b,m,j}^E), \\ \text{s.t. (7b), (7e)}, j = \arg\min_{j\in\mathcal{F}_b^t} c^m(f,j).\end{aligned}$$

The above expression can be written as

$$\omega(\boldsymbol{w}_b^t) = \sum_{m\in\mathcal{M}_b^t} \frac{1}{w_{b,m}^t} \sum_{f\in\mathcal{F}} \frac{\rho_{m,f}^t((1-z_m^{S,t})\kappa_f + z_m^{S,t}\kappa_j)}{\log_2(1+\frac{p_b|h_{b,m}|^2}{\sigma^2})}.$$

According to the Cauchy-Bunyakovsky-Schwarz inequality, we have the upper bound and the lower bound of $\omega(\boldsymbol{w}_b^t)$, expressed as

$$\begin{aligned}&\sum_{m\in\mathcal{M}_b^t} w_{b,m}^t \sum_{m\in\mathcal{M}_b^t} \frac{1}{w_{b,m}^t} \sum_{f\in\mathcal{F}} \frac{\rho_{m,f}^t((1-z_m^{S,t})\kappa_f + z_m^{S,t}\kappa_j)}{\log_2(1+\frac{p_b|h_{b,m}|^2}{\sigma^2})} \\ &\le W_b \sum_{m\in\mathcal{M}_b^t} \frac{1}{w_{b,m}^t} \sum_{f\in\mathcal{F}} \frac{\rho_{m,f}^t((1-z_m^{S,t})\kappa_f + z_m^{S,t}\kappa_j)}{\log_2(1+\frac{p_b|h_{b,m}|^2}{\sigma^2})},\end{aligned}$$

and

$$\begin{aligned}&\sum_{m\in\mathcal{M}_b^t} \sqrt{\sum_{f\in\mathcal{F}} \frac{\rho_{m,f}^t((1-z_m^{S,t})\kappa_f + z_m^{S,t}\kappa_j)}{\log_2(1+\frac{p_b|h_{b,m}|^2}{\sigma^2})}} \\ &\le \sum_{m\in\mathcal{M}_b^t} w_{b,m}^t \sum_{m\in\mathcal{M}_b^t} \frac{1}{w_{b,m}^t} \sum_{f\in\mathcal{F}} \frac{\rho_{m,f}^t((1-z_m^{S,t})\kappa_f + z_m^{S,t}\kappa_j)}{\log_2(1+\frac{p_b|h_{b,m}|^2}{\sigma^2})}.\end{aligned}$$

If and only if that the upper bound and the lower bound are equal, $\omega(\boldsymbol{w}_b^t)$ can reach the minimum value. The equivalent condition for two bounds is as follows

$$\frac{(w_{b,m}^t)^2}{\sum_{f\in\mathcal{F}} \frac{\rho_{m,f}^t((1-z_m^{S,t})\kappa_f + z_m^{S,t}\kappa_j)}{\log_2(1+\frac{p_b|h_{b,m}|^2}{\sigma^2})}} = k,$$

where k is assumed as a constant. Together with above three expressions, we have another expression about k as follows

$$\begin{aligned}&\sum_{m\in\mathcal{M}_b^t} w_{b,m}^t \\ &= \sum_{m\in\mathcal{M}_b^t} \sqrt{k\sum_{f\in\mathcal{F}} \frac{\rho_{m,f}^t((1-z_m^{S,t})\kappa_f + z_m^{S,t}\kappa_j)}{\log_2(1+\frac{p_b|h_{b,m}|^2}{\sigma^2})}} = W_b.\end{aligned}$$

To this end, we combine two expressions about k and have the optimal w_m^{t*}

$$w_{b,m}^{t*} = \frac{W_b\sqrt{\sum_{f\in\mathcal{F}} \frac{\rho_{m,f}^t((1-z_m^{S,t})\kappa_f + z_m^{S,t}\kappa_j)}{\log_2(1+\frac{p_b|h_{b,m}|^2}{\sigma^2})}}}{\sum_{m\in\mathcal{M}_b^t}\sqrt{\sum_{f\in\mathcal{F}} \frac{\rho_{m,f}^t((1-z_m^{S,t})\kappa_f + z_m^{S,t}\kappa_j)}{\log_2(1+\frac{p_b|h_{b,m}|^2}{\sigma^2})}}}.$$

This completes the proof.

C. Proof of Lemma 3

Although the partition function $Z^{\pi_{\text{old}}}(\mathcal{S}^t)$ is intractable in general, it does not contribute to the gradient about π_{new} and can be ignored. Hence, (13) is rewritten as

$$\pi_{new} = \arg\min_{\pi'} D_{\text{KL}}\left(\pi'(\mathcal{S}^t) \parallel \exp\left(\frac{1}{\delta}Q(\mathcal{S}^t)\right)\right).$$

According to the definition of the KL divergence, let $\pi_{old} \in \Pi$ and let π_{new} be defined as

$$\begin{aligned}\pi_{new}(\mathcal{S}^0) &= \arg\min_{\pi'\in\Pi} D_{KL}\left(\pi'(\mathcal{S}^0) \parallel \exp\left(\frac{1}{\delta}Q(\mathcal{S}^0)\right)\right) \\ &= \arg\min_{\pi'\in\Pi} \sum \pi'(\mathcal{S}^0)\log\frac{\pi'(\mathcal{S}^0)}{\exp\left(\frac{1}{\delta}Q(\mathcal{S}^0)\right)} \\ &= \arg\min_{\pi'\in\Pi} \sum \pi'(\mathcal{S}^0)\left(\log\pi'(\mathcal{S}^0) - \frac{1}{\delta}Q(\mathcal{S}^0)\right) \\ &= \arg\max_{\pi'\in\Pi} \sum \frac{\pi'(\mathcal{S}^0)}{\delta}\left(Q(\mathcal{S}^0) - \delta\log\pi'(\mathcal{S}^0)\right) \\ &= \arg\max_{\pi'\in\Pi} \mathbb{E}\left[Q(\mathcal{S}^0) - \delta\log\pi(\mathcal{S}^0)\right] \\ &= \arg\max_{\pi'\in\Pi} \mathbb{E}\left[\sum_t \mathcal{R}^t(\mathcal{S}^t) + \delta\mathcal{H}(\pi(\mathcal{S}^t))|\mathcal{S}^0\right] \\ &= \pi_{max}(\mathcal{S}^0).\end{aligned}$$

We can infer that minimizing the KL divergence is equivalent to the objective function. Interested readers are referred to [47] for more proof procedures in detail. This completes the proof.

REFERENCES

[1] K. Wang, Y. Zhou, Z. Liu, Z. Shao, X. Luo, and Y. Yang, "Online task scheduling and resource allocation for intelligent NOMA-based industrial Internet of Things," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 5, pp. 803–815, May 2020.

[2] N. Jiang, Y. Deng, A. Nallanathan, and J. Yuan, "A decoupled learning strategy for massive access optimization in cellular IoT networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 3, pp. 668–685, Mar. 2021.

[3] T. Zhang, "Toward automated vehicle teleoperation: Vision, opportunities, and challenges," *IEEE Internet Things J.*, vol. 7, no. 12, pp. 11347–11354, Dec. 2020.

[4] Ericsson. (Nov. 2021). *Ericsson Mobility Report*. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/reports/november-2021

[5] Y. Yang, "Multi-tier computing networks for intelligent IoT," *Nature Electron.*, vol. 2, no. 1, pp. 4–5, Jan. 2019.

[6] S. Aram and B. Jabbari, "Downlink performance of multi-tier wireless networks using punctured Poisson process model," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[7] Z. Shen, J. Jin, T. Zhang, A. Tagami, T. Higashino, and Q. Han, "Data-driven edge computing: A fabric for intelligent building energy management systems," *IEEE Ind. Electron. Mag.*, vol. 16, no. 2, pp. 2–10, Nov. 2021.

[8] J. Zhou, Y. Sun, C. Tellambura, and G. Y. Li, "Joint user grouping, sparse beamforming, and subcarrier allocation for D2D underlaid cache-enabled C-RANs with rate splitting," *IEEE Trans. Veh. Technol.*, vol. 71, no. 4, pp. 3792–3806, Apr. 2022.

[9] F. Wang, F. Wang, J. Liu, R. Shea, and L. Sun, "Intelligent video caching at network edge: A multi-agent deep reinforcement learning approach," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Toronto, ON, Canada, Jul. 2020, pp. 2499–2508.

[10] X. Li, X. Wang, P.-J. Wan, Z. Han, and V. C. M. Leung, "Hierarchical edge caching in device-to-device aided mobile networks: Modeling, optimization, and design," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1768–1785, Aug. 2018.

[11] H. Zhu, Y. Cao, W. Wang, T. Jiang, and S. Jin, "Deep reinforcement learning for mobile edge caching: Review, new features, and open issues," *IEEE Netw.*, vol. 32, no. 6, pp. 50–57, Nov. 2018.

[12] M. Ahmed, S. Traverso, P. Giaccone, E. Leonardi, and S. Niccolini, "Analyzing the performance of LRU caches under non-stationary traffic patterns," 2013, *arXiv:1301.4909*.

[13] A. Jaleel, K. B. Theobald, S. C. Steely, and J. Emer, "High performance cache replacement using re-reference interval prediction (RRIP)," in *Proc. 37th Annu. Int. Symp. Comput. Archit.*, Jun. 2010, pp. 60–71.

[14] Y. M. Saputra, D. T. Hoang, D. N. Nguyen, E. Dutkiewicz, D. Niyato, and D. I. Kim, "Distributed deep learning at the edge: A novel proactive and cooperative caching framework for mobile edge networks," *IEEE Wireless Commun. Lett.*, vol. 8, no. 4, pp. 1220–1223, Aug. 2019.

[15] G. Sun, H. Al-Ward, G. O. Boateng, and G. Liu, "Autonomous cache resource slicing and content placement at virtualized mobile edge network," *IEEE Access*, vol. 7, pp. 84727–84743, 2019.

[16] P. Sermpezis, T. Giannakas, T. Spyropoulos, and L. Vigneri, "Soft cache hits: Improving performance through recommendation and delivery of related content," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1300–1313, Jun. 2018.

[17] A. Devlic, P. Kamaraju, P. Lungaro, Z. Segall, and K. Tollmar, "QoE-aware optimization for video delivery and storage," in *Proc. IEEE 16th Int. Symp. A World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Jun. 2015, pp. 1–10.

[18] Y. Fu, Y. Zhang, Q. Zhu, M. Chen, and T. Q. S. Quek, "Joint content caching, recommendation, and transmission optimization for next generation multiple access networks," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 5, pp. 1600–1614, May 2022, doi: 10.1109/JSAC.2022.3146901.

[19] Y. Fu, Z. Yang, T. Q. S. Quek, and H. H. Yang, "Towards cost minimization for wireless caching networks with recommendation and uncharted Users' feature information," *IEEE Trans. Wireless Commun.*, vol. 20, no. 10, pp. 6758–6771, Oct. 2021.

[20] T. Giannakas, P. Sermpezis, and T. Spyropoulos, "Show me the cache: Optimizing cache-friendly recommendations for sequential content access," in *Proc. 19th IEEE WoWMoM*, Chania, Greece, Jun. 2018, pp. 14–22.

[21] T. Giannakas, A. Giovanidis, and T. Spyropoulos, "SOBA: Session optimal MDP-based network friendly recommendations," in *Proc. 40th IEEE INFOCOM Conf. Comput. Commun.*, May 2021, pp. 1–10.

[22] M. Garetto, E. Leonardi, and G. Neglia, "Similarity caching: Theory and algorithms," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Toronto, ON, Canada, Jul. 2020, pp. 526–535.

[23] M. Costantini, T. Spyropoulos, T. Giannakas, and P. Sermpezis, "Approximation guarantees for the joint optimization of caching and recommendation," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, Jun. 2020, pp. 1–7.

[24] A. Sabnis, T. Si Salem, G. Neglia, M. Garetto, E. Leonardi, and R. K. Sitaraman, "GRADES: Gradient descent for similarity caching," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Vancouver, BC, Canada, May 2021, pp. 1–10.

[25] C. A. Gomez-Uribe and N. Hunt, "The Netflix recommender system: Algorithms, business value, and innovation," *ACM Trans. Manage. Inf. Syst.*, vol. 6, no. 4, p. 13, 2016.

[26] R. Zhou, S. Khemmarat, and L. Gao, "The impact of YouTube recommendation system on video views," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas. (IMC)*, Melbourne, VIC, Australia, Nov. 2010, pp. 404–410.

[27] R. Amir, P. Erickson, and J. Jin, "On the microeconomic foundations of linear demand for differentiated products," *J. Econ. Theory*, vol. 169, pp. 641–665, May 2017.

[28] T. Giannakas, T. Spyropoulos, and P. Sermpezis, "The order of things: Position-aware network-friendly recommendations in long viewing sessions," in *Proc. Int. Symp. Model. Optim. Mobile, Ad Hoc, Wireless Netw. (WiOPT)*, Avignon, France, Jun. 2019, pp. 1–8.

[29] M. Costantini and T. Spyropoulos, "Impact of popular content relational structure on joint caching and recommendation policies," in *Proc. 18th WiOPT*, Volos, Greece, Jun. 2020, pp. 1–8.

[30] M. Garetto, E. Leonardi, and G. Neglia, "Content placement in networks of similarity caches," *Comput. Netw.*, vol. 201, Dec. 2021, Art. no. 108570.

[31] D. Tsigkari and T. Spyropoulos, "User-centric optimization of caching and recommendations in edge cache networks," in *Proc. IEEE 21st Int. Symp. World Wireless, Mobile Multimedia Networks (WoWMoM)*, Aug. 2020, pp. 244–253.

[32] D. Tsigkari and T. Spyropoulos, "An approximation algorithm for joint caching and recommendations in cache networks," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 2, pp. 1826–1841, Jun. 2022.

[33] S. Vural, P. Navaratnam, N. Wang, C. Wang, L. Dong, and R. Tafazolli, "In-network caching of Internet-of-Things data," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Sydney, NSW, Australia, Jun. 2014, pp. 3185–3190.

[34] S. Anokye, D. Ayepah-Mensah, A. M. Seid, G. O. Boateng, and G. Sun, "Deep reinforcement learning-based mobility-aware UAV content caching and placement in mobile edge networks," *IEEE Syst. J.*, vol. 16, no. 1, pp. 275–286, Mar. 2022.

[35] C. Zhong, M. C. Gursoy, and S. Velipasalar, "Deep reinforcement learning-based edge caching in wireless networks," *IEEE Trans. Cognit. Commun. Netw.*, vol. 6, no. 1, pp. 48–61, Mar. 2020.

[36] X. Wang, C. Wang, X. Li, V. C. M. Leung, and T. Taleb, "Federated deep reinforcement learning for Internet of Things with decentralized cooperative edge caching," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9441–9455, Oct. 2020.

[37] Z. Ma, N. Nuermaimaiti, H. Zhang, H. Zhou, and A. Nallanathan, "Deployment model and performance analysis of clustered D2D caching networks under cluster-centric caching strategy," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4933–4945, Aug. 2020.

[38] X. Wu, X. Li, J. Li, P. C. Ching, V. C. M. Leung, and H. V. Poor, "Caching transient content for IoT sensing: Multi-agent soft actor-critic," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 5886–5901, Sep. 2021.

[39] M. X. Goemans, L. Li, V. S. Mirrokni, and M. Thottan, "Market sharing games applied to content distribution in ad hoc networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 5, pp. 1020–1033, May 2006.

[40] P. P. Analytis, D. Barkoczi, and S. M. Herzog, "Social learning strategies for matters of taste," *Nature Human Behaviour*, vol. 2, no. 6, pp. 415–424, May 2018.

[41] S. Ren and M. Van Der Schaar, "Distributed power allocation in multi-user multi-channel cellular relay networks," *IEEE Trans. Wireless Commun.*, vol. 9, no. 6, pp. 1952–1964, Jun. 2010.

[42] A. Bulut and T. K. Ralphs, "On the complexity of inverse mixed integer linear optimization," *SIAM J. Optim.*, vol. 31, no. 4, pp. 3014–3043, Nov. 2021.

[43] C. Sun, X. Wu, X. Li, Q. Fan, J. Wen, and V. C. M. Leung, "Cooperative computation offloading for multi-access edge computing in 6G mobile networks via soft actor critic," *IEEE Trans. Netw. Sci. Eng.*, early access, Apr. 30, 2021, doi: 10.1109/TNSE.2021.3076795.

[44] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, p. 19, Jan. 2016.

[45] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," *Relatório técnico, Telecom ParisTech*, vol. 2011, pp. 1–6, Dec. 2011.

[46] Z. Lyu, Y. Wang, M. Liu, and Y. Chen, "Service-driven resource management in vehicular networks based on deep reinforcement learning," in *Proc. IEEE 31st Annu. Int. Symp. Pers., Indoor Mobile Radio Commun.*, London, U.K., Aug. 2020, pp. 1–6.

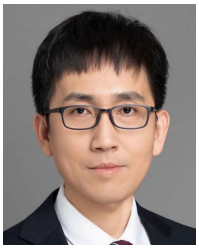
[47] T. Haarnoja et al., "Soft actor-critic algorithms and applications," 2018, *arXiv:1812.05905*.

Chuan Sun (Student Member, IEEE) received the B.S. degree from the Wuhan University of Science and Technology, Wuhan, China, in 2017. He is currently pursuing the Ph.D. degree with the School of Big Data and Software Engineering, Chongqing University, Chongqing, China. His current research interests include multi-access edge computing, recommender systems, and federated learning.

Xiuhua Li (Member, IEEE) received the B.S. degree from the Honors School, Harbin Institute of Technology, China, in 2011, the M.S. degree from the School of Electronics and Information Engineering, Harbin Institute of Technology, in 2013, and the Ph.D. degree from the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada, in 2018. He joined Chongqing University through the One-Hundred Talents Plan of Chongqing University in 2019. He is currently a Tenure-Track Assistant Professor with the School of Big Data and Software Engineering, and also a member of the Haihe Laboratory, ITAI. He is the Head of the Institute of Intelligent

Software and Services Computing associated with the Key Laboratory of Dependable Service Computing in Cyber Physical Society, Chongqing University, Education Ministry, China. His current research interests include 5G/6G mobile internet, mobile edge computing and caching, big data, and machine learning.

Junhao Wen (Member, IEEE) received the Ph.D. degree from Chongqing University in 2008, where he is the Vice Head and a Professor with the School of Big Data and Software Engineering. He has published over 80 refereed journals and conference papers in his research areas. He has over 30 research and industrial grants and developed many commercial systems and software tools. His research interests include service computing, edge computing, and software dependable engineering.



Xiaofei Wang (Senior Member, IEEE) received the B.S. degree from the Huazhong University of Science and Technology, China, and the M.S. and Ph.D. degrees from Seoul National University, Seoul, South Korea. He was a Post-Doctoral Fellow at The University of British Columbia, Vancouver, Canada, from 2014 to 2016. He is currently a Professor with the College of Intelligence and Computing, Tianjin University, Tianjin, China. He has published more than 150 technical papers in IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (JSAC), IEEE TRANSACTIONS ON CLOUD COMPUTING (TCC), IEEE/ACM TRANSACTIONS ON NETWORKING (ToN), IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS (TWC), IEEE INTERNET OF THINGS JOURNAL (IoTJ), IEEE COMMUNICATIONS SURVEYS AND TUTORIALS (COMST), IEEE TRANSACTIONS ON MULTIMEDIA (TMM), INFOCOM, and ICDCS. His research interests include edge computing, edge intelligence, and edge systems, In 2017, he was a recipient of the IEEE ComSoc Fred W. Ellersick Prize and the IEEE ComSoc Asia-Pacific Outstanding Paper Award in 2022.

Zhu Han (Fellow, IEEE) received the B.S. degree in electronic engineering from Tsinghua University in 1997 and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park, in 1999 and 2003, respectively.

From 2000 to 2002, he was the Research and Development Engineer at JDSU, Germantown, MD, USA. From 2003 to 2006, he was a Research Associate at the University of Maryland. From 2006 to 2008, he was an Assistant Professor at Boise State University, ID, USA. Currently, he is a John and Rebecca Moores Professor with the Electrical and Computer Engineering Department and the Computer Science Department, University of Houston, TX, USA. His research interests include wireless resource allocation and management, wireless communications and networking, game theory, big data analysis, security, and smart grid. He was an IEEE Communications Society Distinguished Lecturer from 2015 to 2018. He has been a fellow of AAAS, since 2019, and an ACM Distinguished Member, since 2019. He has been a 1% highly cited researcher according to Web of Science, since 2017. He received the NSF Career Award in 2010, the Fred W. Ellersick Prize of the IEEE Communication Society in 2011, the EURASIP Best Paper Award for the *EURASIP Journal on Advances in Signal Processing* in 2015, the IEEE Leonard G. Abraham Prize in the field of Communications Systems [Best Paper Award in IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (JSAC)] in 2016, and several Best Paper Awards in IEEE conferences. He is also the Winner of the 2021 IEEE Kiyo Tomiyasu Award, for outstanding early to mid-career contributions to technologies holding the promise of innovative applications, with the following citation: "for contributions to game theory and distributed management of autonomous communication networks."

Victor C. M. Leung (Life Fellow, IEEE) is a Distinguished Professor of computer science and software engineering with Shenzhen University, China. He is also an Emeritus Professor of electrical and computer engineering and the Director of the Laboratory for Wireless Networks and Mobile Systems with The University of British Columbia (UBC), Canada. He has published widely in his research areas. His research interests include wireless networks and mobile systems. He is a fellow of the Royal Society of Canada (Academy of Science), the Canadian Academy of Engineering, and the Engineering Institute of Canada. He is named in the current Clarivate Analytics list of Highly Cited Researchers. He received the 1977 APEBC Gold Medal, the NSERC Postgraduate Scholarships from 1977 to 1981, the IEEE Vancouver Section Centennial Award, the 2011 UBC Killam Research Prize, the 2017 Canadian Award for Telecommunications Research, the 2018 IEEE TCGCC Distinguished Technical Achievement Recognition Award, and the 2018 ACM MSWiM Reginald Fessenden Award. He coauthored papers that received the 2017 IEEE ComSoc Fred W. Ellersick Prize, the 2017 IEEE Systems Journal Best Paper Award, the 2018 IEEE CSIM Best Journal Paper Award, and the 2019 IEEE TCGCC Best Journal Paper Award. He is serving on the editorial boards of the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS, IEEE ACCESS, *IEEE Network*, and several other journals.