Proactive Content Caching Based on Actor-Critic Reinforcement Learning for Mobile Edge Networks

Wei Jiang, Daquan Feng, Yao Sun, *Senior Member, IEEE,* Gang Feng, *Senior Member, IEEE,* Zhenzhong Wang, and Xiang-Gen Xia, *Fellow, IEEE*

***Abstract*—Mobile edge caching/computing (MEC) has emerged as a promising approach for addressing the drastic increasing mobile data traffic by bringing high caching and computing capabilities to the edge of networks. Under MEC architecture, content providers (CPs) are allowed to lease some virtual machines (VMs) at MEC servers to proactively cache popular contents for improving users' quality of experience. The scalable cache resource model rises the challenge for determining the ideal number of leased VMs for CPs to obtain the minimum expected downloading delay of users at the lowest caching cost. To address these challenges, in this paper, we propose an actor-critic (AC) reinforcement learning based proactive caching policy for mobile edge networks without the prior knowledge of users' content demand. Specifically, we formulate the proactive caching problem under dynamical users' content demand as a Markov decision process and propose a AC based caching algorithm to minimize the caching cost and the expected downloading delay. Particularly, to reduce the computational complexity, a branching neural network is employed to approximate the policy function in the actor part. Numerical results show that the proposed caching algorithm can significantly reduce the total cost and the average downloading delay when compared with other popular algorithms.**

***Index Terms*—Actor-critic algorithm, Branching neural network, Reinforcement learning, Mobile edge caching.**

I. Introduction

THE development of mobile internet applications is leading an explosive growth of mobile traffic. According to the research from Cisco [1], global mobile data traffic will rise nearly 7-fold from 2016 to 2021. The explosive growth of mobile data traffic will be challenging for the design of next-generation mobile networks. Studies on traffic explosion problem [2], [3] revealed that most data traffic is due to the duplicate downloads of popular content items from remote servers. Exploiting caching and distribution techniques, popular content items are proactively cached in local servers so that requests for the same content items can be fulfilled locally without duplicate transmissions from remote servers [4]. In this way, the redundant traffic can be eliminated and in the meantime users' quality of experience (QoE) can be significantly improved [5].

Content distribution networks (CDN) have been widely used to reduce redundant traffic in wired network paradigm [6]. However, the requested content items must travel through the wireless carrier core network (CN) and radio access network (RAN) before reaching the mobile users. Thus, the wired links accessing to the Internet CDN nodes and the backhaul links between the RAN and CN easily become the bottlenecks that the CDN cannot cope with. To tackle these problems, content items can be cached at the edge of the network rather than CDN so that the content items can be directly delivered to mobile devices, which can significantly offload the traffic flowing to the CDN.

Currently, mobile edge caching/computing (MEC) architecture, which brings network functions and contents to the network edge, has been proposed [7]. In this architecture, mobile edge clouds (*i.e.*, MEC servers) provide a strong computational and storage capacity within RAN that can be used to deploy applications and services as well as to cache popular content items in close proximity to mobile devices [8]. Mobile edge clouds are implemented based on a virtualized platform that leverages recent advancements in network functions virtualization (NFV), information-centric networks (ICN) and software-defined networks (SDN) [9]. Specifically, NFV enables a single mobile edge cloud to create multiple virtual machines (VMs) to provide elastic resource allocation for content providers (CPs) [10]. Since caches in other domains (such as hardware caches, web caches, *etc.*) are typically of a fixed size, the scalable resource model of the cloud gives rise to the challenge for determining the ideal cache size for a CP to obtain maximum benefit (reduced end-to-end latency and database load) at the lowest cost for resource use. Furthermore, mobile edge clouds can perform specific tasks that cannot be fulfilled with the traditional network infrastructure, such as mobile big data analytics, context-aware service performance optimization. Under this circumstance, mobile edge clouds can acquire accurate cell information (such as users' demands, radio conditions, *etc.*) for dynamically performing optimal

This work was supported in part by the National Key R&D Program of China under Grant 2020YFB1807601, and in part by the Innovation Project of Guangdong Educational Department under Grant 2019KTSCX147, and in part by the Shenzhen Science and Technology Program under Grants JCYJ20210324095209025, and in part by ZTE Industry-Academia-Research Cooperation Funds. D. Feng is the corresponding author (fdquan@szu.edu.cn).

W. Jiang and D. Feng are with the Shenzhen Key Laboratory of Digital Creative Technology, the Guangdong Province Engineering Laboratory for Digital Creative Technology, Guangdong Key Laboratory of Intelligent Information Processing, College of Electronics and Information Engineering, Shenzhen University, Shenzhen 518060, China.

Y. Sun is with James Watt School of Engineering, University of Glasgow, G12 8QQ, Scotland, UK.

G. Feng is with the Yangtze Delta Region Institute (Huzhou), University of Electronic Science and Technology of China, Huzhou 313001, China, and also with the National Key Laboratory of Science and Technology on Communications, University of Electronic Science and Technology of China, Chengdu 611731, China.

Z. Wang is with the Technical Management Center, China Media Group, Beijing 100020, China.

X.-G. Xia is with the Department of Electrical and Computer Engineering, University of Delaware, Newark, DE 19716, USA.

content caching [11].

To improve users' QoE, CPs may be willing to lease some VMs and proactively cache popular content items at mobile edge clouds. Most existing studies on proactive caching assume the content popularity or users' content demand is known in advance [12]–[16]. Although some studies assume that the content popularity, which may reflect the average preferences of a large group of users, can be known through statistics, it cannot reflect the preference of each individual user. In reality, the users' preference and content demand are dynamically varying in spatial and temporal domains [17]. This inspires us to exploit machine learning methods to learn the users' content demand and decide what to cache at mobile edge clouds. Although some studies have used machine learning methods to proactive caching [18]–[24], the number of leased VMs is always assumed fixed. In fact, the number of leased VMs will affect the performance of cache-enabled mobile edge networks. Since the users' content demand is dynamically varying in spatial and temporal domains, CPs would like to smartly change the number of leased VMs to save the caching cost while guarantee the users' QoE. Therefore, it is meaningful to dynamically optimize the number of leased VMs with the adjustment of the caching policy to realize the tradeoff between the caching cost and the average downloading delay under dynamical users' content demand.

In this paper, we propose a novel proactive caching algorithm for mobile edge networks to minimize the caching cost as well as the expected downloading delay of users. First, instead of assuming a priori knowledge of users' content demand, which might be externally given in advance or estimated in a separate training phase, our proposed algorithm learns the users' content demand online by observing the historical content demand for cached content items in each mobile edge cloud. Second, different from the existing works which assume a fixed cache size, to the best of our knowledge, our work is the first work to consider the scalable cache resource model, which optimizes the number of leased VMs with the adjustment of the caching policy to realize the trade-off between the caching cost and the average downloading delay under dynamical users' content demand. Third, in order to overcome the computationally intensive issue with large action spaces in the action-value methods and high variance issue of evaluating a policy in the policy gradient methods, an actor-critic (AC) reinforcement learning based algorithm is proposed to solve the proactive caching problem in a long time period. Fourth, since the action space is too large and thus the actor needs an exponential number of times to update the policy function, a branching neural network is employed to estimate the policy function, thus to effectively reduce the computational complexity. Specifically, the main contributions of this paper are summarized as follows:

- We investigate the proactive caching problem for mobile edge networks without assuming the prior knowledge of users' content demand. Meanwhile, we consider a scalable cache resource model and dynamically adjust the number of leased VMs to match the dynamical users' content demand, thus to realize the trade-off between the caching cost and the average downloading delay of users.
- We model the proactive caching problem under dynamical users' content demand as a Markov decision process (MDP) and propose an AC reinforcement learning based caching algorithm to derive the caching solution with aim of foresightedly minimizing the caching cost as well as the expected downloading delay of users. Neural networks (NNs) are employed to estimate the state-value function and policy function in the critic and actor, respectively. Particularly, in the actor part, as our model has high-dimensional action space and the policy update requires tackling the combinatorial increase of the number of action dimensions, we employ a novel neural architecture with multiple network branches, one for each action dimension, to approximate the policy function. This approach achieves a linear increase of the number of network outputs by allowing a level of independence for each individual action dimension.
- We conduct a series of experiments with different system parameters and a real world dataset to verify the effectiveness of our proposed AC based algorithm. It is shown that the proposed AC based caching algorithm can significantly reduce the total cost as well as the average downloading delay of users and improve the cache hit rate when compared with other state-of-the-art caching algorithms.

The remainder of this paper is organized as follows. We introduce the related work and the system model in Section II and Section III, respectively. In Section IV, we present the MDP model for formulating the proactive caching problem. An actor-critic reinforcement learning based caching algorithm is proposed to solve the problem in Section V. In Section VI, we evaluate the performance of the proposed algorithm by simulation. Finally, we conclude this paper in Section VII.

II. Related Work

In recent years, the design of proactive caching policy for mobile edge networks has attracted intensive research interest. Existing mobile edge caching policies broadly fall into two categories: those with perfect knowledge of users' content demand and those without.

A. Caching Policies with Perfect Knowledge of Users' Content Demand

Thus far, most relevant studies assume that either users' instantaneous content demand or content popularity distribution are perfectly known in advance. K. Shanmugam *et al.* [12] proposed a local caching policy for small cells aiming to minimize the average download delay with known users' content demand distribution. In [13], the authors proposed a content popularity based caching scheme to maximize the cache hit ratio, *i.e.*, each base station (BS) caches the most popular content items independently. In [14], a collaborative edge caching strategy was studied for MEC networks to improve caching resource utilization. The authors of [15] proposed a joint content caching and delivering policy with assuming the content popularity known in advance. In [16], the authors investigated cooperative caching problem with aim of providing a global optimal caching solution for heterogeneous networks by assuming the arrival intensity of users' requests and content popularity known a priori. However, these studies are not suitable to be applied when the users' content demand or content popularity are time-varying.

B. Caching Policies without Perfect Knowledge of Users' Content Demand

There is little research work in this category. A comparison of the characteristics of our proposed AC reinforcement learning based proactive caching algorithm with other similar learning approaches in the literature is summarised in Table I. Driven by a proactive caching paradigm, in [18] and [19], the authors proposed transfer learning based approaches to estimate the content popularity by leveraging user content correlations and users' historical content demand. While those transfer learning based approaches require training sets of known content popularity, our proposed algorithm does not need a training phase, but learns the users' content demand online, and thus adapts to a varying users' content demand.

The authors of [20] proposed a multi-armed bandit (MAB) algorithm to learn the content popularity distribution online by refreshing the cached content and observing instantaneous content demand for cached content items. Some extensions of the MAB learning algorithm were proposed in [21] and [22]. The authors of [21] exploited the learned content popularity to optimize the cache policy by taking the users' connectivity to the small base stations (SBSs) into consideration. In [22], the authors investigated the content caching and sharing problem from an MAB learning perspective, where the learning of the content popularity distribution is incorporated with the content caching and sharing process. The approaches in [21] and [22] assume a specific type of content popularity distribution. The assumption is restrictive as in practice the type of content popularity distribution is unknown in advance. In contrast, our proposed algorithm is model-free since it does not assume a specific type of content popularity distribution. Moreover, in [20]–[22], the MAB learning algorithm needs to estimate the action values and find the action with the highest expected reward. However, to achieve this, it needs to resort to an optimization procedure in every state encountered to find the action that leads to an optimal value. Therefore, those action-value methods can be computationally intensive and may not be suitable if the state space and action space are large.

The authors of [23] modelled the proactive caching problem as a Markov decision process to minimize the expected long-term average cost, and proposed low complexity parameterized policy representations and used policy gradient reinforcement learning to optimize the parameters. However, the policy gradient methods suffer from high variance in the estimates of the gradient, leading to slow learning. In [24], a deep reinforcement learning framework was proposed for edge caching to maximize the cache hit rate, when the content popularity information is unknown and only the historical users' content demand can be observed. Deep reinforcement learning based predictions are effective, but require huge training data. Moreover, the cache size is typically fixed in these studies. Taking the caching cost into consideration, it is a necessary and challenging work to intelligently vary the number of leased VMs with the dynamical users' content demand, thus to obtain the maximum cache benefit at the lowest cost.

III. System Model

The system model of proactive caching for mobile edge networks is illustrated in Fig. 1. We consider a cache-enabled mobile edge network with N mobile edge clouds which are co-located in the BSs [9]. We denote $\mathcal{MC} = \{1, 2, ..., N\}$ as the set of mobile edge clouds and $\mathcal{CC} = \{1, 2, ..., N, N+1\}$ as the set of clouds, where cloud $N+1$ is the central cloud. In addition, we assume there exists a cloud operation controller, which can timely know the content demands in these clouds at current stage and deploy content items in any mobile edge clouds at the next stage in a centralized way [25]. Operators can open the mobile edge clouds to CPs, allowing them to acquire the content demand information and lease some resources (*e.g.*, storage, bandwidth, computation capacity) of the mobile edge clouds to improve users' QoE [7].

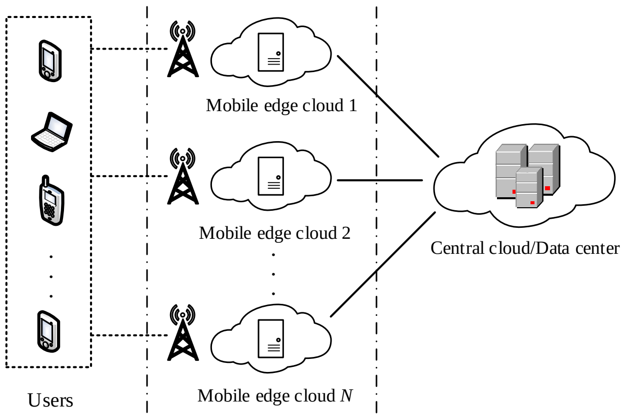


Fig. 1. System model of proactive caching for mobile edge networks.

We assume that there exists a set of users $\mathcal{U} = \{1, 2, ..., U\}$ and each user is connected to one or more mobile edge clouds. Let $\mathcal{F} = \{1, 2, ..., F\}$ be the set of content items provided by the CP. Each content item $f \in \mathcal{F}$ is of size s_f. We assume that users' content request occurs independently following Poisson point processes with average arrival rate λ_u. The user's preference is defined as $p_{u,f}$, *i.e.*, the probability of content item f requested by user u when the user has a content request, and $\sum_{f\in\mathcal{F}} p_{u,f} = 1$.

Users can request content items from their connected clouds. We define a binary variable $y_{u,i,f} \in \{0, 1\}$ as a user's content request association indicator to indicate whether user u requests content item f from cloud i or not: $y_{u,i,f} = 1$ if user u requests content item f from cloud i and $y_{u,i,f} = 0$ otherwise. The users' content request association matrix is denoted as $\mathbf{Y} = [y_{u,i,f}]_{u\in\mathcal{U},f\in\mathcal{F},i\in\mathcal{CC}}$. Hence, the users' demand for content item f in cloud i can be obtained, which is defined

TABLE I
COMPARISON WITH RELATED WORK ON LEARNING-BASED PROACTIVE CACHING

Work	[18], [19]	[20]	[21], [22]	[23]	[24]	This Work
Model-Free	Yes	Yes	No	Yes	Yes	Yes
Online/Offline-Learning	Offline	Online	Online	Online	Online	Online
Free of Training Phase	No	Yes	Yes	Yes	No	Yes
Value Function-Based	No	Yes	Yes	No	Yes	Yes
Policy-Based	No	No	No	Yes	Yes	Yes
Scalable Cache Resource Model	No	No	No	No	No	Yes

as $d_{i,f} = \sum_{u\in\mathcal{U}} y_{u,i,f} \cdot \lambda_u \cdot p_{u,f}$. The users' content demand matrix is denoted as $\mathbf{S} = [d_{i,f}]_{i\in\mathcal{CC}, f\in\mathcal{F}}$.

Users' content request association indicators are exceptionally dependent of the caching decision. We define another binary variable $a_{i,f} \in \{0,1\}$ as a caching decision indicator to indicate whether caching content item f in mobile edge cloud i or not: $a_{i,f} = 1$ if caching content item f in mobile edge cloud i and $a_{i,f} = 0$ otherwise. The caching decision matrix is defined as $\mathbf{A} = [a_{i,f}]_{i\in\mathcal{MC}, f\in\mathcal{F}}$. User u can make a request for content item f from cloud i only when content item f is cached in cloud i. Therefore, we have $y_{u,i,f} \le a_{i,f}$. We assume that the central cloud has all content items. This assumption is justified by the fact that the central cloud can retrieve any content item from the Internet.

Let the data transmission rate of cloud i to user u be denoted as $c_{u,i}$. The system load for a cloud can be defined as the fraction of time required to deliver traffic loads from the cloud to users. The system load for cloud i can be represented as $\rho_i = \sum_{u\in\mathcal{U}} \sum_{f\in\mathcal{F}} y_{u,i,f} \cdot \lambda_u \cdot p_{u,f} \cdot s_f / c_{u,i}$, and $\rho_i < 1$ for system stability. Let $\boldsymbol{\rho} = [\rho_i]_{i\in\mathcal{CC}}$ be the system load vector.

On one hand, the more content items are cached in mobile edge clouds, the more cache resources are needed, and the larger caching cost is incurred. Hence, we define the caching cost as the total number of cached content items, which is given by $C_c = \sum_{i\in\mathcal{MC}} \sum_{f\in\mathcal{F}} a_{i,f}$. On the other hand, in order to minimize the average downloading delay of users, we introduce a delay-optimal metric to demonstrate the downloading performance. We assume that the queuing model of the system is M/G/s [26]. Thus, the number of downloading flows in cloud i is $\rho_i/(1-\rho_i)$. The delay-optimal performance function can be formulated as $C_d = \sum_{i\in\mathcal{CC}} \rho_i/(1-\rho_i)$. Note that C_d equals to the number of downloading flows in the system. According to Little's law, if we try to minimize C_d, it is indeed equivalent to minimizing the average downloading delay [27]. In order to minimize the caching cost as well as the average downloading delay of users, in this paper, we define the total cost function as:

$$C = C_c + \xi C_d, \tag{1}$$

where ξ is a delay equivalent cost parameter indicating the equivalent cost for one downloading flow waiting in the system and reflecting the importance of the delay performance relative to the caching cost. Here, we define a reward function as:

$$R = -C. \tag{2}$$

Apparently, maximizing the expected reward equals to minimizing the expected total cost.

To demonstrate the temporal users' content demand variations, in this paper, we assume that time is slotted into periods. The superscript k is used to represent the k-th time period. In each time period, the CP makes a caching decision according to the historical users' content demand with aim to maximize the expected reward, namely to minimize the expected total cost. In the next section, we model the proactive caching problem for mobile edge networks as an MDP.

IV. MDP FRAMEWORK FOR PROACTIVE CACHING IN MOBILE EDGE NETWORKS

MDP is a classical formalization of sequential decision making. In each time period, the agent observes the current state and selects an action, and then interacts with the environment to obtain certain reward and proceed to the next state. The problem is to decide which action should be selected based on the current information, so that the discounted accumulative reward in a long-term can be maximized.

In the caching decision process for mobile edge clouds, in time period k, when the users' content demand matrix is $\mathbf{S}^{(k)}$, the CP selects caching decision matrix $\mathbf{A}^{(k)}$ according to a strategy π. The users correspondingly send their content requests to the clouds according to users' content request association matrix $\mathbf{Y}^{(k)}$, which can be determined by the cached content items and the queuing delay in each cloud. Thereafter, as the users' content requests occur, the users' content demand matrix transits into $\mathbf{S}^{(k+1)}$, which is determined by the varying number of users' content requests associated to each cloud in time period k. Meanwhile, the immediate reward $R^{(k)}(\mathbf{S}^{(k)}, \mathbf{A}^{(k)})$ calculated by Eqs. (1) and (2) is fed back to the CP. Apparently, the caching decision process for mobile edge clouds is an MDP, where a caching decision matrix and a users' content demand matrix correspond to an action and a state, respectively. The MDP framework for proactive caching in mobile edge networks is illustrated in Fig. 2.

Formally, an MDP is defined as a tuple $E = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$. $\mathcal{S} = \{s\}$ is the state space, where element s is a state; $\mathcal{A} = \{a\}$ is the action space, where element a is an action; $\mathcal{P} = \{P(s'|s,a)\}$ is a state transition probability function, where element $P(s'|s,a)$ represents the probability that state transits into s' by taking action a under state s; $\mathcal{R} = \{R(s,a)\}$ is a reward function, where element $R(s,a)$ represents the obtained reward by taking action a under state s; $\gamma \in (0,1]$ is a discount factor which weighs the relative contribution of current and future rewards.

In the MDP framework for proactive caching in mobile edge clouds, the state space is denoted as $\mathcal{S} = \{\mathbf{S}\}$, which represents the set of users' content demand matrices. The action space is denoted as $\mathcal{A} = \{\mathbf{A}\}$, which represents the set of caching decision matrices. In time

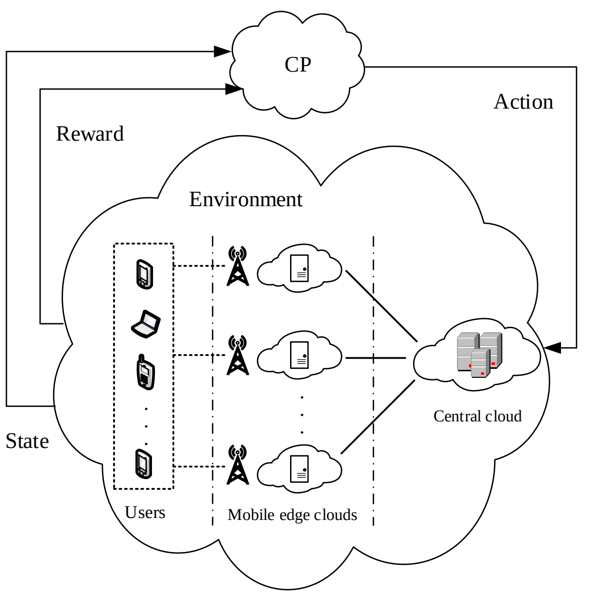


Fig. 2. The MDP framework for proactive caching in mobile edge networks.

period k, action $\mathbf{A}^{(k)}$ is selected in state $\mathbf{S}^{(k)}$, and then reward $R^{(k)}(\mathbf{S}^{(k)},\mathbf{A}^{(k)})$ is obtained while the state transits into $\mathbf{S}^{(k+1)}$. The state transition probability is denoted as $P(\mathbf{S}'|\mathbf{S},\mathbf{A}) = Pr\left\{\mathbf{S}^{(k+1)}=\mathbf{S}'|\mathbf{S}^{(k)}=\mathbf{S},\mathbf{A}^{(k)}=\mathbf{A}\right\}$. Note that the state transition probabilities are unknown in our proactive caching problem due to the dynamical users' content demand.

The agent selects an action in a state according to a policy function π which refers to the mapping relationship between state space and action space. For a stochastic policy, represented as $\pi(a|s)$, the probability of taking action a in state s is $\pi(a|s)$. The state value function is introduced to evaluate the performance of a policy. The state value function $V^{\pi}(s)$ is defined as the discounted accumulative reward obtained by selecting actions according to policy π starting from state s, namely:

$$V^{\pi}(s)=\mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k}R^{(k)}(s^{(k)},a^{(k)}|s^{(0)}=s)\right], \tag{3}$$

where $s^{(0)}$ is the starting state; $R^{(k)}(s^{(k)},a^{(k)})$ is the obtained reward in time period k by taking action $a^{(k)}$ in state $s^{(k)}$.

In the MDP framework for proactive caching in mobile edge clouds, the goal of the CP is to find a policy π, which maps a state $\mathbf{S}$ to an action $\mathbf{A}$ with probability $\pi(\mathbf{A}|\mathbf{S})$ to maximize the discounted accumulative reward starting from state $\mathbf{S}$. As mentioned before, this discounted accumulative reward is called as a state value function, which can be represented by:

$$\begin{aligned}V^{\pi}(\mathbf{S})&=\mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k}R^{(k)}(\mathbf{S}^{(k)},\mathbf{A}^{(k)}|\mathbf{S}^{(0)}=\mathbf{S})\right]\\&=\sum_{\mathbf{A}\in\mathcal{A}}\pi(\mathbf{A}|\mathbf{S})\left[R^{(k)}(\mathbf{S},\mathbf{A})+\gamma\sum_{\mathbf{S}'\in\mathcal{S}}P(\mathbf{S}'|\mathbf{S},\mathbf{A})V^{\pi}(\mathbf{S}')\right].\end{aligned} \tag{4}$$

The optimal policy π^* satisfies the Bellman equation [28]:

$$\begin{aligned}&V^{*}(\mathbf{S})=V^{\pi^*}(\mathbf{S})=max\\&\left\{\sum_{\mathbf{A}\in\mathcal{A}}\pi^{*}(\mathbf{A}|\mathbf{S})\left[R^{(k)}(\mathbf{S},\mathbf{A})+\gamma\sum_{\mathbf{S}'\in\mathcal{S}}P(\mathbf{S}'|\mathbf{S},\mathbf{A})V^{\pi^*}(\mathbf{S}')\right]\right\}.\end{aligned} \tag{5}$$

There are some well-known approaches to solve the MDP problems such as dynamic programming [29]. Unfortunately, these approaches depend on perfect knowledge of the dynamic variation of environment to a great extent. However, it is challenging to have the prior knowledge of the future users' content demand. Hence, in the next section, we employ a reinforcement learning approach to solve the MDP problem without requiring the information of users' content demand in advance.

V. Actor-Critic Reinforcement Learning based Caching Policy

There are mainly three types of reinforcement learning algorithms: actor-only, critic-only, and actor-critic, where actor and critic correspond to the policy and value function respectively [30]. Although actor-only methods (such as policy gradient methods) can learn stochastic policies effectively for the model with large action space and converge to a local optimum asymptotically, they usually lead to high variance in the estimates of expected rewards and learn slowly. Critic-only methods (such as action-value methods) usually use temporal difference (TD) iteration and thus have a lower variance in the estimates of expected rewards, but they need to resort to an optimization procedure in every state encountered for finding the action with the highest expected reward, and thus they usually cannot efficiently resolve the problems with large action spaces, which is the case in our proposed problem. To improve the learning performance, we thus employ the actor-critic methods which combine the strong aspects of actor-only and critic-only algorithms. The actor is capable of producing discrete or continuous actions without the need for optimization procedures on a value function. The critic provides the estimated value function to the actor for updating the policy parameters with lower variance [26]. These properties of actor-critic algorithms have made them a preferred reinforcement learning algorithm to solve the MDP problem in this paper.

Fig. 3 illustrates the actor-critic learning framework. The agent can be divided into two entities: actor (policy) and critic (value function). At a given state, the actor selects an action according to the policy. After interacting with the environment, the state transits into a new one while certain reward is obtained. Then, the critic evaluates the quality of the current policy and updates the value function by exploiting a TD error. After that, the actor will update the policy by using the information from the critic. The actor-critic algorithm repeats the above process until it coverages.

Based on the actor-critic learning framework, in the following, we propose an actor-critic based algorithm for solving the proactive caching problem in mobile edge clouds. Specifically, NNs are employed to estimate the state-value function and

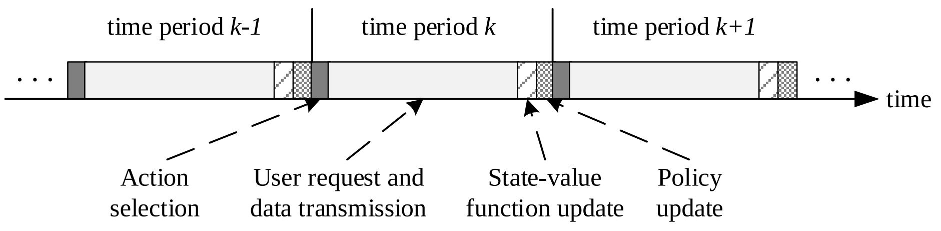


Fig. 4. Illustration of AC reinforcement learning framework for proactive caching.

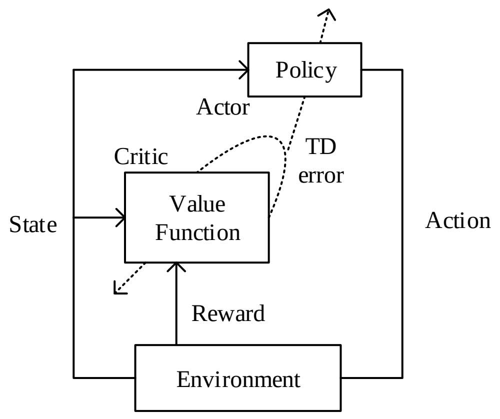


Fig. 3. Actor-critic learning framework.

policy function in the critic and actor, respectively. The illustration of AC reinforcement learning framework for proactive caching is shown in Fig. 4.

Action selection: We assume that the system is at the beginning of time period k. Meanwhile, the current state is $\mathbf{S}^{(k)}$. Thereafter, the CP needs to select an action according to a stochastic policy aiming at maximizing the accumulated expected reward in a long-term. There is a well-known tradeoff between exploration and exploitation: searching for a better action (exploration) and achieving the highest empirical rewards (exploitation). One of the most common methodologies is to employ a Boltzmann distribution as a stochastic policy. The CP selects action $\mathbf{A}$ in state $\mathbf{S}^{(k)}$ of time period k with probability

$$\pi(\mathbf{A}|\mathbf{S}^{(k)}) = \frac{\exp\{h(\mathbf{S}^{(k)},\mathbf{A})/\tau\}}{\sum\limits_{\mathbf{A}'\in\mathcal{A}}\exp\{h(\mathbf{S}^{(k)},\mathbf{A}')/\tau\}}. \tag{6}$$

where τ is an adjustment parameter with a positive value, and $h(\mathbf{S}^{(k)},\mathbf{A})$ is an action selection preference function which indicates the preference to select action $\mathbf{A}$ at state $\mathbf{S}^{(k)}$.

User request and data transmission: In time period k, after the CP caches some content items in mobile edge clouds according to the selected action $\mathbf{A}^{(k)}$, each mobile edge cloud broadcasts the cached content items and current system load $\rho_i^{(k)}$ to users. When the users' content requests occur, they correspondingly send their content requests to the clouds according to users' content request association matrix $\mathbf{Y}^{(k)}$, which can be determined by the cached content items, system loads and data transmission rates of each mobile edge cloud. Formally, when user u has a content request for content item f in time period k, the user's content request association indicator is defined as

$$y_{u,i,f}^{(k)} = \begin{cases} 1, \ i = \arg\max\limits_{i\in\mathcal{CC}} \frac{c_{u,i}}{\left(1-\rho_i^{(k)}\right)^{-2}} \cdot a_{i,f}^{(k)}, \\ \quad 0, \ otherwise. \end{cases} \tag{7}$$

In [31], Eq. (7) has been proved to be the optimal to achieve the minimum of the average download delay if the caching decision matrix is determined. Note that when the system loads of each cloud $\rho_i^{(k)}$, $i \in \mathcal{CC}$ are the same, the users will send their content requests to the cloud which has cached the requested content item while having the highest data transmission rate.

State-value function update: Since the state-value function cannot be calculated for infinite state problems with the Bellman equation, in this paper, we employ an NN for estimating the state-value function. As shown in Fig. 5, we put the vector $\Psi(\mathbf{S}) = (\psi_1(\mathbf{S}), \psi_2(\mathbf{S}), ..., \psi_d(\mathbf{S}))$, which is called *feature vector* representing state $\mathbf{S}$, into an NN, where d is the number of features of state $\mathbf{S}$. The output is the approximate state-value function $V(\mathbf{S},\boldsymbol{w})$, where $\boldsymbol{w} \in \mathbb{R}^d$ is the parameter vector and can be updated by using gradient descent method. Since the performance of linear feature-based NN is sufficiently good for our scenario with low complexity and it provides a good convergence and stability, we apply a linear feature-based NN to approximate the state-value function. As the linear feature-based NN cannot take into account of any interactions between features, it needs instead, or in addition, features for combinations of all underlying states. Hence, in this paper, we employ a polynomial method [32] to construct the features of state $\mathbf{S}$ as the input of NN. Since the delay-optimal performance function C_d (*i.e.*, the number of downloading flows in the system) is only related to the total numbers of users' content requests in every cloud, let $\mathbf{x}(\mathbf{S}) = (x_1(\mathbf{S}), x_2(\mathbf{S}), ..., x_{N+1}(\mathbf{S}))$ be the basis feature vector, where $x_i(\mathbf{S}) = \sum_{f\in\mathcal{F}} s_{i,f}$, $i \in \mathcal{CC}$ (*i.e.*, the total numbers of users' content requests in every cloud) are the basis features. Then, each polynomial-basis feature $\psi_i(\mathbf{S})$ can be written as

$$\psi_i(\mathbf{S}) = \prod_{j=1}^{N+1} (x_j(\mathbf{S}))^{n_{i,j}}, \tag{8}$$

where each $n_{i,j}$ is an integer in the set $\{0,1\}$. These polynomial-basis features form the feature vector $\Psi(\mathbf{S})$ which contains $d = 2^{N+1}$ different features. Then, the output approximate state-value function $V(\mathbf{S}, \boldsymbol{w})$ can be represented as

$$V(\mathbf{S}, \boldsymbol{w}) = \boldsymbol{w}^T \cdot \Psi(\mathbf{S}) = \sum_{i=1}^{d} w_i \cdot \psi_i(\mathbf{S}). \tag{9}$$

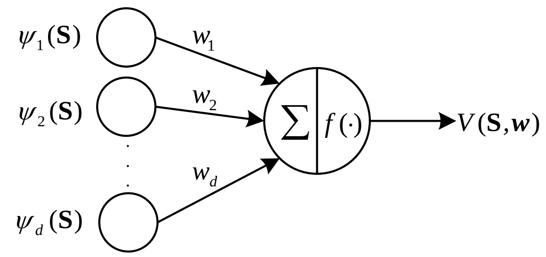


Fig. 5. A neural network for estimating the state-value function.

After the user request and data transmission phase in time period k, the users' content demand in each cloud may change, and thus the system state transits into $\mathbf{S}^{(k+1)}$. Meanwhile, the reward $R^{(k)}(\mathbf{S}^{(k)}, \mathbf{A}^{(k)})$ of selecting action $\mathbf{A}^{(k)}$ at state $\mathbf{S}^{(k)}$ in time period k can be calculated by Eqs. (1) and (2). Consequently, the critic computes TD error $\delta^{(k)}(\mathbf{S}^{(k)}, \mathbf{A}^{(k)})$ by exploiting the approximate state-value function and obtained reward. Specifically, TD error $\delta^{(k)}(\mathbf{S}^{(k)}, \mathbf{A}^{(k)})$ is the difference between $R^{(k)}(\mathbf{S}^{(k)}, \mathbf{A}^{(k)}) + \gamma \cdot V^{(k)}(\mathbf{S}^{(k+1)}, \boldsymbol{w})$ and the state-value function $V^{(k)}(\mathbf{S}^{(k)}, \boldsymbol{w})$ estimated at the preceding state, namely,

$$\begin{aligned}\delta^{(k)}(\mathbf{S}^{(k)}, \mathbf{A}^{(k)}) =& R^{(k)}(\mathbf{S}^{(k)}, \mathbf{A}^{(k)}) + \gamma \cdot V^{(k)}(\mathbf{S}^{(k+1)}, \boldsymbol{w}) \\ & - V^{(k)}(\mathbf{S}^{(k)}, \boldsymbol{w}).\end{aligned} \tag{10}$$

Afterwards, the critic would feed the TD error back to the actor. Meanwhile, the critic exploits the TD error to update the parameter vector $\boldsymbol{w}$ by gradient descent method. Specifically, the parameter vector $\boldsymbol{w}$ is updated by

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha_{\boldsymbol{w}}^{(k)} \delta^{(k)}(\mathbf{S}^{(k)}, \mathbf{A}^{(k)}) \nabla_{\boldsymbol{w}} V(\mathbf{S}^{(k)}, \boldsymbol{w}), \tag{11}$$

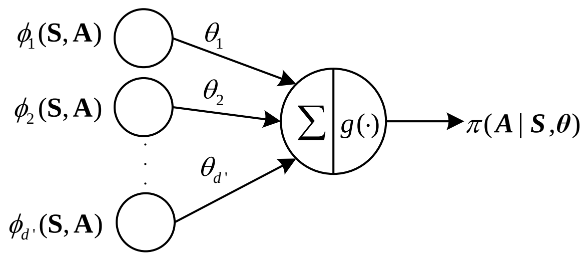
where $\alpha_{\boldsymbol{w}}^{(k)}$ is a positive step-size parameter which affects the convergence rate, and $\nabla_{\boldsymbol{w}} V(\mathbf{S}^{(k)}, \boldsymbol{w})$ gives the direction of the gradient. Since the NN is linear feature-based, according to (9), we have $\nabla_{\boldsymbol{w}} V(\mathbf{S}^{(k)}, \boldsymbol{w}) = \Psi^{(k)}(\mathbf{S})$.

Policy update: Similar to the estimation of the state-value function, we use another NN to obtain the approximate policy function π, as shown in Fig. 6. We put the feature vector $\Phi(\mathbf{S}, \mathbf{A}) = (\phi_1(\mathbf{S}, \mathbf{A}), \phi_2(\mathbf{S}, \mathbf{A}), ..., \phi_{d'}(\mathbf{S}, \mathbf{A}))$ which represents state $\mathbf{S}$ and action $\mathbf{A}$ into the NN in Fig. 6, where d' represents the number of features of state $\mathbf{S}$ and action $\mathbf{A}$. The output is the approximate policy function $\pi(\mathbf{A}|\mathbf{S}, \boldsymbol{\theta})$, represented by

$$\pi(\mathbf{A}|\mathbf{S}, \boldsymbol{\theta}) = \frac{\exp\{h(\mathbf{S}, \mathbf{A}, \boldsymbol{\theta})/\tau\}}{\sum\limits_{\mathbf{A}' \in \mathcal{A}} \exp\{h(\mathbf{S}, \mathbf{A}', \boldsymbol{\theta})/\tau\}}, \tag{12}$$

where $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ is the parameter vector of the NN, and the action selection preference function is given as

$$h(\mathbf{S}, \mathbf{A}, \boldsymbol{\theta}) = \boldsymbol{\theta}^T \cdot \Phi(\mathbf{S}, \mathbf{A}) = \sum_{i=1}^{d'} \theta_i \cdot \phi_i(\mathbf{S}, \mathbf{A}). \tag{13}$$



$\phi_1(\mathbf{S},\mathbf{A})$
$\phi_2(\mathbf{S},\mathbf{A})$
$\phi_{d'}(\mathbf{S},\mathbf{A})$
θ_1
θ_2
$\theta_{d'}$
$\sum$
$g(\cdot)$
$\pi(A\,|\,S,\theta)$

Fig. 6. A neural network for estimating the policy function.

At the end of time period k, the actor would use the TD error to update the policy, *i.e.*, updating the parameter vector $\boldsymbol{\theta}$. The parameter vector $\boldsymbol{\theta}$ is updated by

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}}^{(k)} \delta^{(k)}(\mathbf{S}^{(k)}, \mathbf{A}^{(k)}) \nabla_{\boldsymbol{\theta}} \ln \pi(\mathbf{A}^{(k)}|\mathbf{S}^{(k)}, \boldsymbol{\theta}), \tag{14}$$

where $\alpha_{\boldsymbol{\theta}}^{(k)}$ is a positive step-size parameter that affects the convergence rate, and $\nabla_{\boldsymbol{\theta}} \ln \pi(\mathbf{A}^{(k)}|\mathbf{S}^{(k)}, \boldsymbol{\theta})$ gives the direction of the gradient. According to (12) and (13), we have

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \ln \pi(\mathbf{A}^{(k)}|\mathbf{S}^{(k)}, \boldsymbol{\theta}) =& \Phi(\mathbf{S}^{(k)}, \mathbf{A}^{(k)}) \\ & - \sum_{\mathbf{A}' \in \mathcal{A}} \pi(\mathbf{A}'|\mathbf{S}^{(k)}, \boldsymbol{\theta}) \cdot \Phi(\mathbf{S}^{(k)}, \mathbf{A}').\end{aligned} \tag{15}$$

However, the action space is too large (*i.e.*, $|\mathcal{A}| = 2^{(N+1)\cdot F}$) and the computational complexity of policy update is exponential to the number of mobile edge clouds $N+1$ and the number of content items F. These properties make policy update hard to implement. In order to reduce the computational complexity, in the following, we employ a novel neural architecture with several network branches, which is called *branching neural network*, to estimate the policy function. In a branching neural network for the policy function estimation, action space can be separated into several dimensions and each neural network branch estimates a policy function on an action dimension. In [33], A. Tavakoli *et al.* have proven the effectiveness of the branching neural network model.

As shown in Fig. 7, we separate action $\mathbf{A}$ into F dimensions, *i.e.*, $\mathbf{A} = (\mathbf{A}_1{}^T, \mathbf{A}_2{}^T, ..., \mathbf{A}_F{}^T)$ and $\mathbf{A}_f = [a_{i,f}]_{i\in\mathcal{MC}}$, $f \in \mathcal{F}$. The action space of the f-th dimension is denoted as $\mathcal{A}_f = \{\mathbf{A}_f\}$. Hence, we need F neural network branches to estimate the policy function in each action dimension. Specifically, the output of the f-th neural network branch is represented by

$$\pi(\mathbf{A}_f|\mathbf{S}, \boldsymbol{\theta}_f) = \frac{\exp\{h(\mathbf{S}, \mathbf{A}_f, \boldsymbol{\theta}_f)/\tau\}}{\sum\limits_{\mathbf{A}_f{}' \in \mathcal{A}_f} \exp\{h(\mathbf{S}, \mathbf{A}_f{}', \boldsymbol{\theta}_f)/\tau\}}, \tag{16}$$

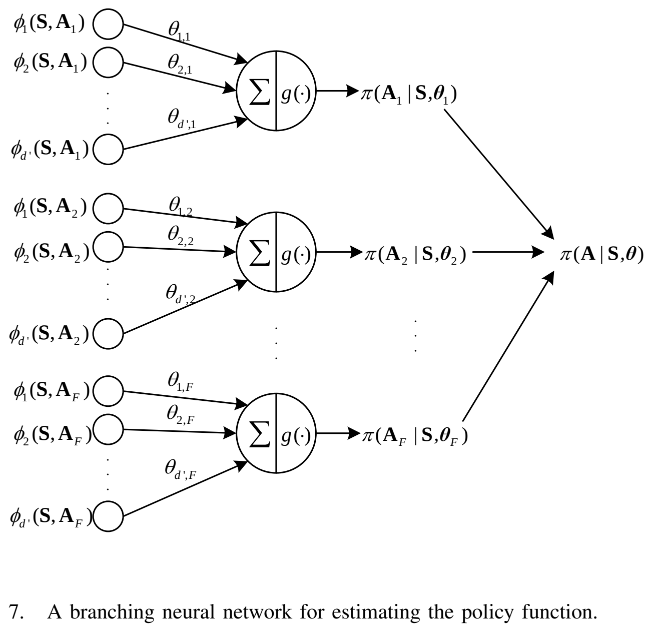
where $\boldsymbol{\theta}_f \in \mathbb{R}^{d'}$ is the parameter vector of the f-th neural network branch, and the action selection preference function of the f-th action dimension is given by

$$h(\mathbf{S}, \mathbf{A}_f, \boldsymbol{\theta}_f) = \boldsymbol{\theta}_f{}^T \cdot \Phi(\mathbf{S}, \mathbf{A}_f) = \sum_{i=1}^{d'} \theta_{i,f} \cdot \phi_i(\mathbf{S}, \mathbf{A}_f). \tag{17}$$

where $\Phi(\mathbf{S}, \mathbf{A}_f) = (\phi_1(\mathbf{S}, \mathbf{A}_f), \phi_2(\mathbf{S}, \mathbf{A}_f), ..., \phi_{d'}(\mathbf{S}, \mathbf{A}_f))$ is the feature vector which represents state $\mathbf{S}$ and action $\mathbf{A}_f$. We also use the polynomial method [32] to construct the features of state $\mathbf{S}$ and action $\mathbf{A}_f$. Each polynomial-basis feature $\phi_i(\mathbf{S}, \mathbf{A}_f)$ is denoted as

$$\phi_i(\mathbf{S}, \mathbf{A}_f) = \sum_{i\in\mathcal{CC}} s_{i,f} \cdot \prod_{j=1}^{N} (a_{j,f})^{n_{i,j}}, \tag{18}$$

where $\sum_{i\in\mathcal{CC}} s_{i,f}$ represents the total number of users' content requests for content item f. These polynomial-basis features form the feature vector $\Phi(\mathbf{S}, \mathbf{A}_f)$ which contains $d' = 2^N$ different features.



$\phi_1(\mathbf{S},\mathbf{A}_1)$
$\phi_2(\mathbf{S},\mathbf{A}_1)$
$\phi_{d'}(\mathbf{S},\mathbf{A}_1)$
$\theta_{1,1}$
$\theta_{2,1}$
$\theta_{d',1}$
$\sum$
$g(\cdot)$
$\pi(\mathbf{A}_1|\mathbf{S},\boldsymbol{\theta}_1)$
$\phi_1(\mathbf{S},\mathbf{A}_2)$
$\phi_2(\mathbf{S},\mathbf{A}_2)$
$\phi_{d'}(\mathbf{S},\mathbf{A}_2)$
$\theta_{1,2}$
$\theta_{2,2}$
$\theta_{d',2}$
$\pi(\mathbf{A}_2|\mathbf{S},\boldsymbol{\theta}_2)$
$\pi(\mathbf{A}|\mathbf{S},\boldsymbol{\theta})$
$\phi_1(\mathbf{S},\mathbf{A}_F)$
$\phi_2(\mathbf{S},\mathbf{A}_F)$
$\phi_{d'}(\mathbf{S},\mathbf{A}_F)$
$\theta_{1,F}$
$\theta_{2,F}$
$\theta_{d',F}$
$\pi(\mathbf{A}_F|\mathbf{S},\boldsymbol{\theta}_F)$

Fig. 7. A branching neural network for estimating the policy function.

At the end of time period k, the actor uses the TD error to update the parameter vector of the branching neural network. The parameter vector $\boldsymbol{\theta}_f$, $f \in \mathcal{F}$ is updated by

$$\boldsymbol{\theta}_f \leftarrow \boldsymbol{\theta}_f + \alpha_{\boldsymbol{\theta}_f}^{(k)} \delta^{(k)}(\mathbf{S}^{(k)}, \mathbf{A}^{(k)}) \nabla_{\boldsymbol{\theta}_f} \ln \pi(\mathbf{A}_f{}^{(k)}|\mathbf{S}^{(k)}, \boldsymbol{\theta}_f), \tag{19}$$

where $\alpha_{\boldsymbol{\theta}_f}^{(k)}$ is a positive step-size parameter that affects the convergence rate, and $\nabla_{\boldsymbol{\theta}_f} \ln \pi(\mathbf{A}_f{}^{(k)}|\mathbf{S}^{(k)}, \boldsymbol{\theta}_f)$ gives the direction of the gradient. According to (16) and (17), we have

$$\begin{aligned}\nabla_{\boldsymbol{\theta}_f} \ln \pi(\mathbf{A}_f{}^{(k)}|\mathbf{S}^{(k)}, \boldsymbol{\theta}_f) &= \Phi(\mathbf{S}^{(k)}, \mathbf{A}_f{}^{(k)}) \\ &- \sum_{\mathbf{A}_f{}' \in \mathcal{A}_f} \pi(\mathbf{A}_f{}'|\mathbf{S}^{(k)}, \boldsymbol{\theta}_f) \cdot \Phi(\mathbf{S}^{(k)}, \mathbf{A}_f{}').\end{aligned} \tag{20}$$

In policy update, the number of actions in each dimension is $|\mathcal{A}_f| = 2^{(N+1)}$, $f \in \mathcal{F}$ and the computational complexity of updating the parameter vector of the branching neural network is exponential to the number of mobile edge clouds $N+1$ and linear to the number of content items F. Although each neural network branch ignores the influence of other action space dimensions when learning the policy function in one action dimension, it greatly reduces the computational complexity of updating the parameter vector of the branching neural network, making it feasible to use neural network estimating policy function in the case of large action space. The AC reinforcement learning based caching policy is summarized in Algorithm 1.

Algorithm 1

```
1: Initialize vectors $\boldsymbol{w}$, $\boldsymbol{\theta}_f$, $f \in \mathcal{F}$.
2: for each time period $k = 0, 1, 2, ...$ do
3:     Action selection: Select action $\mathbf{A}_f{}^{(k)}$ in state $\mathbf{S}^{(k)}$ according to policy $\pi(\mathbf{A}_f|\mathbf{S}, \theta_f)$, $f \in \mathcal{F}$ in (16).
4:     User request and data transmission: When user $u$ has a content request for content item $f$, he/she send his/her content request to the cloud according to the user's content request association indicator in (7).
5:     State-value function update: (a) Observe the next state $\mathbf{S}^{(k+1)}$; (b) Calculate reward $R^{(k)}(\mathbf{S}^{(k)}, \mathbf{A}^{(k)})$ according to (1) and (2); (c) Put feature vectors $\Psi(\mathbf{S}^{(k)})$ and $\Psi(\mathbf{S}^{(k+1)})$ into the NN to compute the state value function $V(\mathbf{S}^{(k)}, \boldsymbol{w})$ and $V(\mathbf{S}^{(k+1)}, \boldsymbol{w})$ respectively; (d) Calculate TD error $\delta^{(k)}(\mathbf{S}^{(k)}, \mathbf{A}^{(k)})$ according to (10); (e) Update parameter vector $\boldsymbol{w}$ by (11).
6:     Policy update: (a) Put feature vector $\Phi(\mathbf{S}, \mathbf{A}_f)$ into the branching neural network to compute policy function $\pi(\mathbf{A}_f|\mathbf{S}, \boldsymbol{\theta}_f)$, $\mathbf{A}_f \in \mathcal{A}_f$, $f \in \mathcal{F}$; (b) Update parameter vector $\boldsymbol{\theta}_f$, $f \in \mathcal{F}$ by (19).
7: end for
```

Next, we analyze the convergence of Algorithm 1. In Algorithm 1, state-value function parameter vector $\boldsymbol{w}$ is updated by the critic, while policy parameter vector $\boldsymbol{\theta}_f$, $f \in \mathcal{F}$ is updated by the actor according to the TD error from the critic feedback. The optimal state-value function parameter vector and policy parameter vector would be found when they converge to an optimum state-value function and an optimum policy, respectively. To guarantee the convergence theoretically, the critic's estimation should be at least asymptotically accurate, and thus the step-size parameters $\alpha_{\boldsymbol{w}}^{(k)}$ and $\alpha_{\boldsymbol{\theta}_f}^{(k)}$ should be non-increasing, and typically satisfy the following conditions [34]:

$$\sum_{k=0}^{\infty} \alpha_{\boldsymbol{w}}^{(k)} = \infty,\ \sum_{k=0}^{\infty} \left[\alpha_{\boldsymbol{w}}^{(k)}\right]^2 < \infty,\ \sum_{k=0}^{\infty} \alpha_{\boldsymbol{\theta}_f}^{(k)} = \infty,$$
$$\sum_{k=0}^{\infty} \left[\alpha_{\boldsymbol{\theta}_f}^{(k)}\right]^2 < \infty, \text{ and } \lim_{k\to\infty} \frac{\alpha_{\boldsymbol{\theta}_f}^{(k)}}{\alpha_{\boldsymbol{w}}^{(k)}} = 0.$$

The first four conditions guarantee that the learning process will slow down gradually but never stop. The last condition ensures that the update of the actor operates on a slower time-scale compared to the critic so that the critic has enough time to evaluate the current policy.

VI. Performance Evaluations

In this section, we evaluate the performance of our proposed AC based caching algorithm. The performance metrics we employ include the total cost, the average downloading delay and the cache hit rate. The total cost is defined as $\sum_k C^{(k)}$, where $C^{(k)}$ is calculated by Eq. (1). The average downloading delay is defined as the average delay experienced by all users for downloading requested content items, where the delay of user u downloading content item f from cloud i is denoted by $(s_f/c_{u,i})/(1-\rho_i)$. The cache hit rate is defined as H/R, where H is the number of users' content requests which are satisfied by content retrieval from mobile edge clouds and R is the total number of users' content requests.

We conduct simulation experiments to compare the performance of the proposed AC based caching algorithm with the following four caching algorithms: 1) Informed upper bound (IUB) caching algorithm [35], which assumes that the users' content demand is perfectly known in advance and the CP finds the optimal caching solution by greedily caching as many as content items into mobile edge clouds. 2) ε-greedy caching algorithm [26], which employs ε-greedy instead of Boltzmann distribution as a stochastic policy. 3) State-of-the-art (SOTA) caching algorithm [20], in which each mobile edge cloud employs the modified combinatorial upper confidence bound (MCUCB) algorithm to learn the users' content demand via a combinatorial multi-armed bandit (MAB) framework and caches content items with a greedy approach. 4) Least frequently used (LFU) caching algorithm [36], which assumes that the users' content demand is perfectly known in advance and each mobile edge cloud always keeps the most frequently requested content items in the cache. Modified least recently used (MLRU) caching algorithm [36], which keeps all the content items that have been requested within the last time period and randomly replaces the rest of the content items. 4) Randomized replacement (RR) caching algorithm [37], which randomly caches content items in mobile edge clouds.

We consider a cache-enabled mobile edge network with five mobile edge clouds which are co-located in the BSs [35]. There are 1000 users randomly connected to one or more mobile edge clouds. The content set provided by the CP has 1000 content items, and each content item is of size 10 units [16]. Each user generates content requests independently following Poisson process with average rate 0.1 (arrivals/time period) [17]. The content popularity generally follows ZipF distribution [12], *i.e.*, the probability that a request is for the j-th most popular content item is $j^{-\beta}/\sum_{i=1}^{F} i^{-\beta}$, where the ZipF distribution parameter β is 0.56 [12]. We normalize the data transmission rate from central cloud to users as 1 unit/s and assume the data transmission rate from mobile edge clouds to users is 3 unit/s [38]–[40]. In reality, the transmission rates from central cloud and mobile edge clouds to the end users depend on the transmission distance (or transmission hops) [41]. Due to the long transmission distance (two or more transmission hops) between the central cloud and the end users, it is a reasonable assumption. The delay equivalent cost parameter ξ is 5000 [26]. The discount factor γ is 0.99 [30]. As for the proposed AC based caching algorithm, the step-size parameters $\alpha_{\boldsymbol{w}}^{(k)}$ and $\alpha_{\boldsymbol{\theta}_f}^{(k)}$ are $1/k$ and $1/(k \log k)$, respectively [26]. In LFU, MLRU and RR caching algorithms, we assume that the leased cache size of each mobile edge cloud is fixed to 100 units. Note that the total leased cache size in the cell is 5% of the total content size, which is a realistic assumption [42]. For other caching algorithms, there is no limitation on leased cache size, and the leased cache sizes depend on the cache decisions. The duration of each time period is normalized to one hour. For each experiment, we run the simulation for 300 time periods [23], *i.e.*, each user generates $300 \cdot \lambda_u$ content requests according to the Zipf distribution. To examine the impact of some parameters on the performance, we simulate by adjusting only one parameter while configuring the others according to Table II.

TABLE II
System Parameters

Parameters	Values
Number of mobile edge clouds (N)	5
Number of users (U)	1000
Number of content items (F)	1000
Size of each content item (s_f)	10 units
Average arrival rate of users' content requests (λ_u)	0.1
ZipF distribution parameter (β)	0.56
Data transmission rate of central cloud to users ($c_{u,N+1}$)	1 unit/s
Data transmission rate of mobile edge cloud i to users ($c_{u,i}$)	3 unit/s
Delay equivalent cost parameter (ξ)	5000
Discount factor (γ)	0.99

In the first experiment, we compare the total cost, the average downloading delay and the cache hit rate when the average arrival rate of users' content requests λ_u varies from 0.06 to 0.14 in Fig. 8. We can see that the total cost and the average downloading delay increase with λ_u for the LFU, MLRU and RR caching algorithms in Fig. 8 (a) and (b), respectively. This is because that in the LFU, MLRU and RR caching algorithms, the leased cache size of each mobile edge cloud is fixed. With the increase of λ_u, the system load in each cloud increases, and thus the average downloading delay and the total cost increase accordingly. We can also observe that the total cost and the cache hit rate increase with λ_u for the IUB, AC based, ε-greedy and SOTA caching algorithms in Fig. 8 (a) and (c), respectively, while the average downloading delay decreases with λ_u for the IUB, AC based, ε-greedy and SOTA caching algorithms in Fig. 8 (b), since the IUB, AC based, ε-greedy and SOTA caching algorithms would like to cache more content items in mobile edge clouds to offload the traffic of the central cloud when λ_u increases. As expected, IUB provides an upper bound on the performance of the caching algorithms. Note that since the users' content demand is usually unknown in practice, it is impossible to achieve such performance upper bound. Among the other caching algorithms, the AC based, ε-greedy and SOTA caching algorithms outperform the LFU, MLRU and RR caching algorithms. This is due to the fact that the AC based, ε-greedy and SOTA caching algorithms learn from the historical users' content demand to make the caching decisions for different arrival rates of users' content requests, while RR ignores the history information of users' content demand and MLRU learns the historical users' content demand only from one-step past. Although the LFU caching algorithm utilizes the history information, *i.e.*, the frequency, it is also short of the benefits from cooperative caching. Note that the AC based and ε-greedy caching algorithms outperform SOTA since each mobile edge cloud learns the users' content demand from its own set of connected users without taking into account the overall network connectivity in SOTA, and the AC based caching algorithm outperforms the ε-greedy caching algorithm because of the insufficient exploration issue in the ε-greedy algorithm [32]. In Fig. 8 (a), the total cost of the AC based caching algorithm is significantly lower than that of the ε-greedy, SOTA, LFU, MLRU and RR caching algorithms. In particular, compared with the ε-greedy, SOTA, LFU, MLRU and RR caching algorithms, the gain of the AC based caching algorithm in terms of the total cost is approximately 3%, 5%, 22%, 26% and 29%, respectively, when λ_u is up to 0.14.

Next, we compare the total cost, the average downloading delay and the cache hit rate for different delay equivalent cost parameters in Fig. 9. We can observe that the total cost and the cache hit rate increase with ξ for the IUB, AC based, ε-greedy and SOTA caching algorithms in Fig. 9 (a) and (c), respectively, and the average downloading delay decreases with ξ for the IUB, AC based, ε-greedy and SOTA caching algorithms in Fig. 9 (b). The reasons are as follows. When ξ is large, the delay equivalent cost ξC_d is more significant than caching cost C_c . Therefore, the IUB, AC based, ε-greedy and SOTA caching algorithms would like to cache more content items in mobile edge clouds to reduce the average downloading delay. In Fig. 9 (a), it is shown that the AC based caching algorithm outperforms the ε-greedy, SOTA, LFU, MLRU and RR caching algorithms with the improvement on the total cost approximately 3%, 4%, 16%, 19% and 21%, respectively, when the delay equivalent cost parameter is 7000.

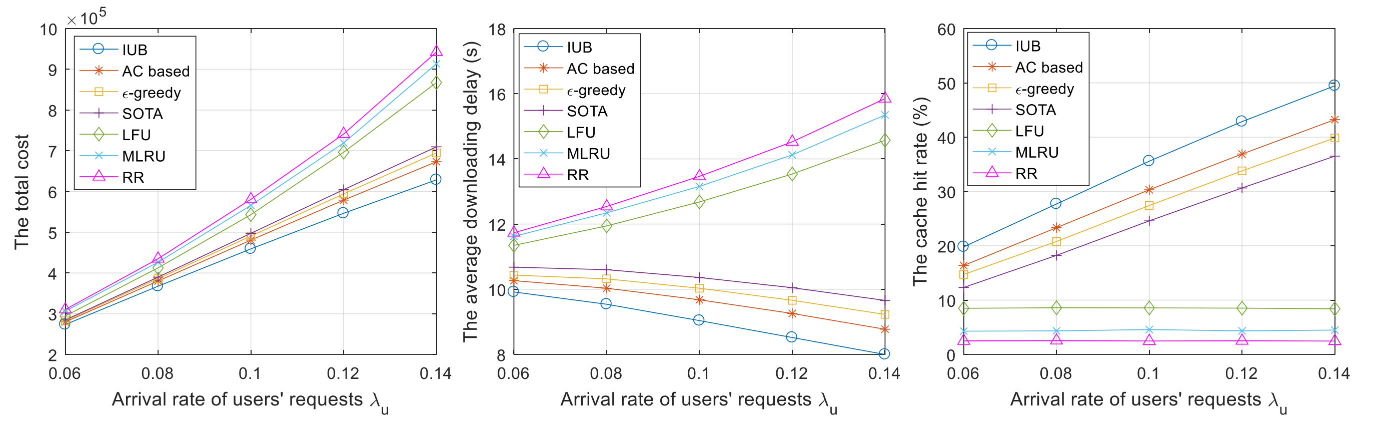
Fig. 10 shows the total cost, the average downloading delay and the cache hit rate as a function of the total number of content items. We can see that the total cost and the average downloading delay increase with the number of content items for all caching algorithms in Fig. 10 (a) and (b), respectively, and the cache hit rate decreases with the number of content items for all caching algorithms in Fig. 10 (c), since more content requests have to be satisfied by central cloud with the increase of F. In Fig. 10 (b), it is shown that the AC based caching algorithm outperforms the ε-greedy, SOTA, LFU, MLRU and RR caching algorithms with the improvement on the average downloading delay approximately 2%, 4%, 12%, 15% and 17%, respectively, when the number of content items is 2500.

Next, we compare the total cost, the average downloading delay and the cache hit rate for varying number of mobile edge clouds from 3 to 7 in Fig. 11. From Fig. 11 (b) and (c), we can observe that the average downloading delay decreases and the cache hit rate increases with N for all caching algorithms since more users' content requests can be served in mobile edge clouds with the increase of N. In Fig. 11 (a), we can see that the total cost decreases with N for the IUB, AC based, ε-greedy, SOTA, LFU and MLRU caching algorithms, while the total cost increases with N for RR. This is because that the increase of caching cost C_c is greater that the decrease of the delay cost with the number of mobile edge clouds for the RR caching algorithm. In Fig. 11 (b), the average downloading delay of the AC based caching algorithm is significantly lower than that of the ε-greedy, SOTA, LFU, MLRU and RR caching algorithms. Compared with the ε-greedy, SOTA, LFU, MLRU and RR caching algorithms, the gain of the AC based caching algorithm in terms of average downloading delay is approximately 4%, 8%, 24%, 27% and 29%, respectively, when the number of mobile edge clouds is 7.

In the following, we examine the impact of the ZipF distribution parameter β on the total cost, the average downloading delay and the cache hit rate in Fig. 12. From Fig. 12, we can see that when β ranges from 0.2 to 1, the total cost as well as the average downloading delay decrease and the cache hit rate increases for all caching algorithms. The reasons are as follows. When β is large, the vast majority of user requests are concentrated on a small number of content items. Clearly, caching the most popular content items provides more significant benefits.

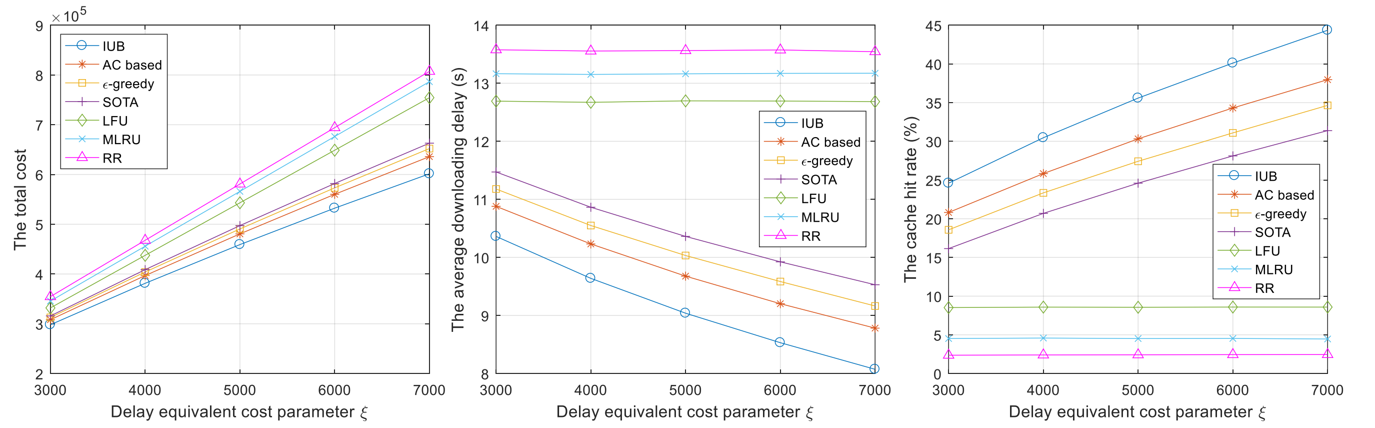
Finally, we use a dataset of MovieLens from the real world [43] to further evaluate the performance of our proposed caching algorithm. The MovieLens 100K Dataset [44] records 943 MovieLens users' rating data in the form of $\langle$User ID, Movie ID, Rating, Timestamp$\rangle$, which contains 100,000 ratings of 1,682 movies during the seven-month period from September 19, 1997 through April 22, 1998. We assume that the movie rating process in the dataset is corresponding to the content request process [45]. We divide the timestamp into time periods of one hour each, and we run the simulation for 300 time periods. Fig. 13 shows the number of content requests in each time period.

We run all caching algorithms on the dataset and study their results as a function of time. Fig. 14(a), 14(b) and 14(c) show the cumulative total cost, the cumulative downloading delay of users and the cumulative numbers of cache hits up to time period t as a function of time, respectively. We can see that for all caching algorithms, the cumulative total cost, the cumulative downloading delay of users and the cumulative numbers of cache hits are increasing abruptly, due to the burstiness of the content request process. As expected, the IUB gives an upper bound to the other caching algorithms. However, since the users' content demand is usually unknown in practice, it is impossible to achieve such performance upper bound. Among the other caching algorithms, our proposed AC based caching algorithm clearly outperforms the ε-greedy,



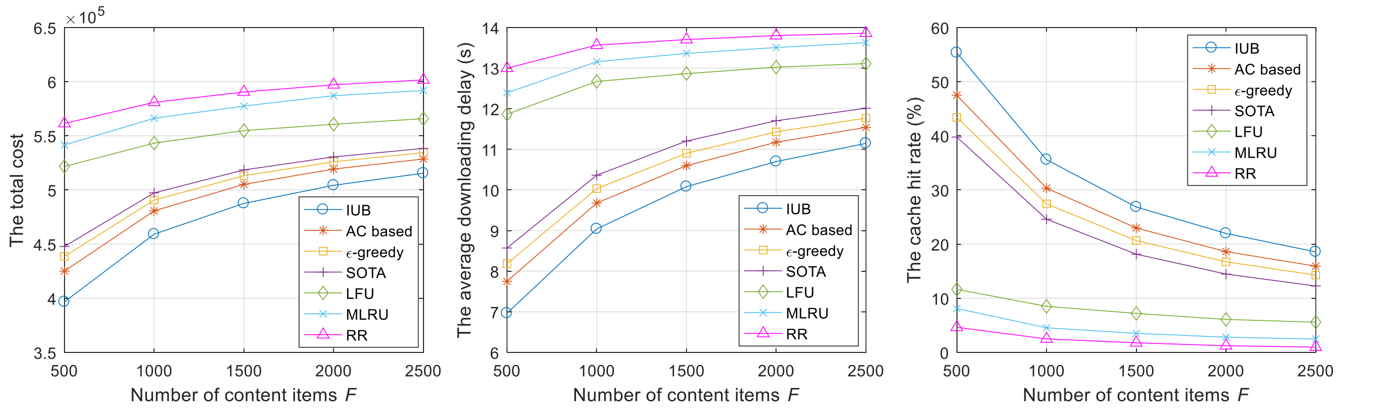
(a) The total cost (b)The average downloading delay (c) The cache hit rate

Fig. 8. The total cost, the average downloading delay and the cache hit rate of the IUB, AC based, ϵ-greedy, SOTA, LFU, MLRU and RR caching algorithms for different arrival rates of users' content requests.



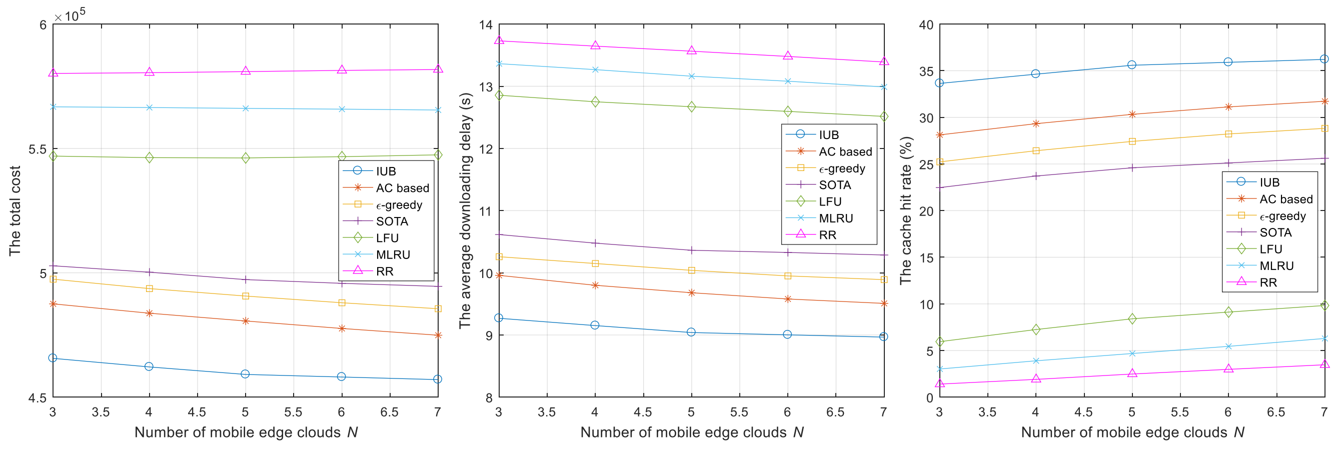
(a) The total cost (b)The average downloading delay (c) The cache hit rate

Fig. 9. The total cost, the average downloading delay and the cache hit rate of the IUB, AC based, ϵ-greedy, SOTA, LFU, MLRU and RR caching algorithms for different balancing parameters.



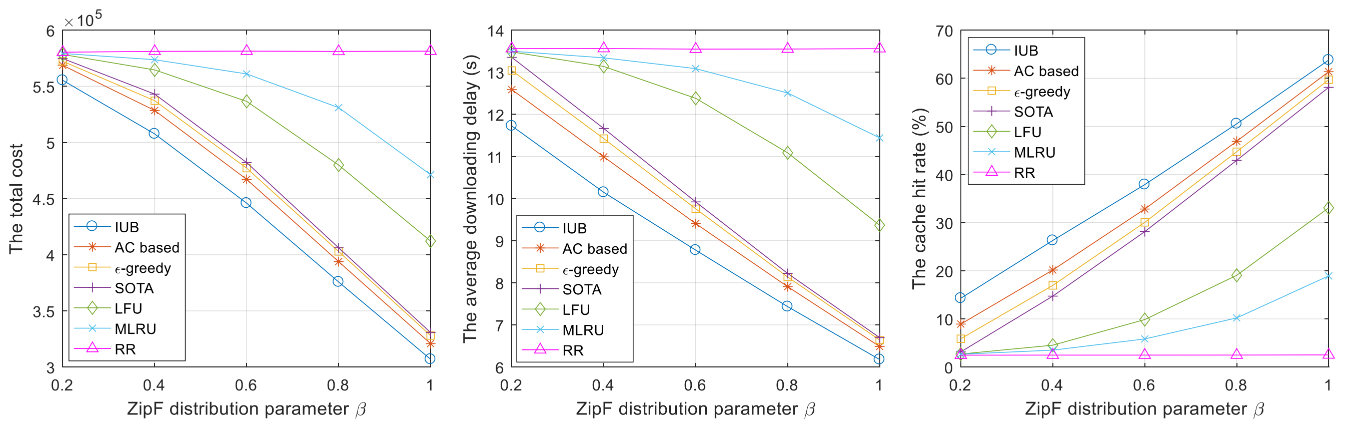
(a) The total cost (b)The average downloading delay (c) The cache hit rate

Fig. 10. The total cost, the average downloading delay and the cache hit rate of the IUB, AC based, ϵ-greedy, SOTA, LFU, MLRU and RR caching algorithms for different numbers of content items.



(a) The total cost (b)The average downloading delay (c) The cache hit rate

Fig. 11. The total cost, the average downloading delay and the cache hit rate of the IUB, AC based, ε-greedy, SOTA, LFU, MLRU and RR caching algorithms for different numbers of mobile edge clouds.



(a) The total cost (b)The average downloading delay (c) The cache hit rate

Fig. 12. The total cost, the average downloading delay and the cache hit rate of the IUB, AC based, ε-greedy, SOTA, LFU, MLRU and RR caching algorithms for different ZipF distribution parameters.

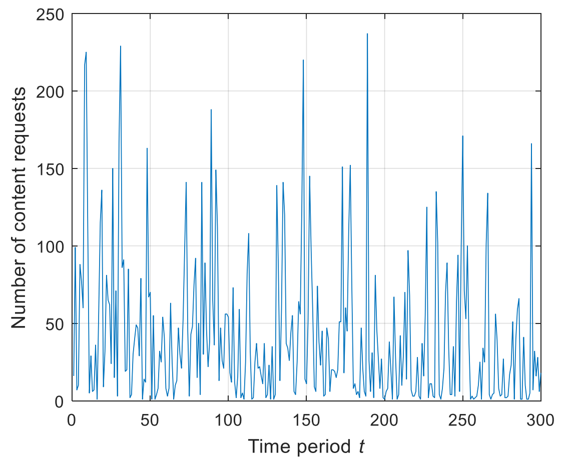


Fig. 13. The number of content requests in each time period.

SOTA, LFU, MLRU and RR caching algorithms. This is due to the fact that the AC based caching algorithm can efficiently learn from the historical users' content demand to make the caching decisions. In particular, compared with the ε-greedy, SOTA, LFU, MLRU and RR caching algorithms, the gain of the AC based caching algorithm in terms of the cumulative total cost up to time period 300 is approximately 1%, 2%, 7%, 9% and 10%, respectively, and the gain of the AC based caching algorithm in terms of the cumulative downloading delay of users up to time period 300 is approximately 3%, 4%, 15%, 18% and 19%, respectively.

VII. Conclusion

In this paper, we have addressed the proactive caching problem for mobile edge networks from a reinforcement learning perspective when the prior information of users' content demand is unknown. Considering the scalable cache resource model of the mobile edge clouds, we specifically formulate the proactive caching problem under dynamical users' content demand as a Markov decision process. Afterwards, we propose

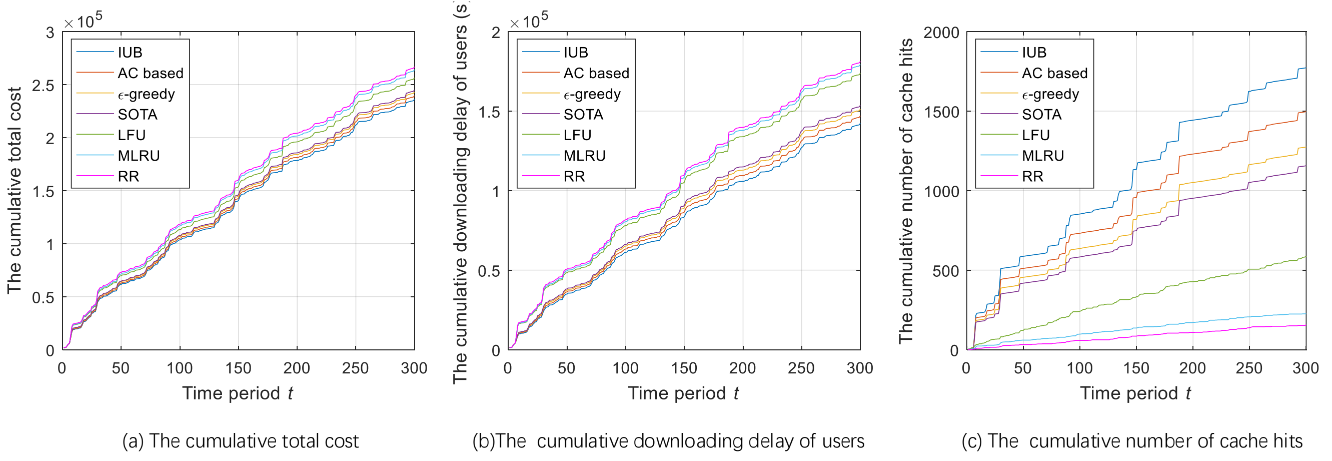


Fig. 14. The cumulative total cost, the cumulative downloading delay of users and the cumulative numbers of cache hits of the IUB, AC based, ε-greedy, SOTA, LFU, MLRU and RR caching algorithms in each time period.

an actor-critic (AC) reinforcement learning based algorithm to give the caching solution to minimize the caching cost as well as the expected downloading delay. Particularly, since the action space is too large and thus the actor needs exponential number of times to update the policy function, a branching neural network is employed to estimate the policy function thus to effectively reduce the computational complexity. Numerical results show that the proposed AC based caching algorithm outperforms other popular caching algorithms in terms of the total cost, the average downloading delay and the cache hit rate.

References

[1] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2016-2021," http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html, 2017.

[2] J. Erman, A. Gerber, M. Hajiaghayi, D. Pei, S. Sen, and O. Spatscheck, "To cache or not to cache: The 3G case," *IEEE Internet Computing*, vol. 15, no. 2, pp. 27–34, 2011.

[3] L. Qiu and G. Cao, "Popularity-aware caching increases the capacity of wireless networks," in *Computer Communications (INFOCOM), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–9.

[4] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. Leung, "Cache in the air: exploiting content caching and delivery techniques for 5G systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 131–139, 2014.

[5] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Communications Magazine*, vol. 8, no. 52, pp. 82–89, 2014.

[6] A. O. Al-Abbasi, V. Aggarwal, and R. Moo-Ryong, "Multi-tier caching analysis in CDN-Based over-the-top video streaming systems," *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 835–847, 2019.

[7] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing–A key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[8] M. Chen, Y. Qian, Y. Hao, Y. Li, and J. Song, "Data-driven computing and caching in 5G networks: Architecture and delay analysis," *IEEE Wireless Communications*, vol. 25, no. 1, pp. 70–75, 2018.

[9] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 54–61, 2017.

[10] H. Liu, F. Eldarrat, H. Alqahtani, A. Reznik, X. de Foy, and Y. Zhang, "Mobile edge cloud system: Architectures, challenges, and approaches," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2495–2508, 2018.

[11] M. T. Beck, M. Werner, S. Feld, and S. Schimper, "Mobile edge computing: A taxonomy," in *Proc. of the Sixth International Conference on Advances in Future Internet*. Citeseer, 2014, pp. 48–55.

[12] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[13] J. Gong, S. Zhou, Z. Zhou, and Z. Niu, "Policy optimization for content push via energy harvesting small cells in heterogeneous networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 2, pp. 717–729, 2017.

[14] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Cooperative content caching in 5G networks with mobile edge computing," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 80–87, 2018.

[15] M. Gregori, J. Gmez-Vilardeb, J. Matamoros, and D. Gndz, "Wireless content caching for small cell and d2d networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1222–1234, 2016.

[16] W. Jiang, G. Feng, and S. Qin, "Optimal cooperative content caching and delivery policy for heterogeneous cellular networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 5, pp. 1382–1393, 2017.

[17] B. Bharath, K. Nagananda, and H. V. Poor, "A learning-based approach to caching in heterogenous small cell networks," *IEEE Transactions on Communications*, vol. 64, no. 4, pp. 1674–1686, 2016.

[18] E. Baştuğ, M. Bennis, and M. Debbah, "A transfer learning approach for cache-enabled wireless networks," in *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), 2015 13th International Symposium on*. IEEE, 2015, pp. 161–166.

[19] T. Hou, G. Feng, S. Qin, and W. Jiang, "Proactive content caching by exploiting transfer learning for mobile edge computing," *International Journal of Communication Systems*, vol. 31, no. 11, p. e3706, 2018.

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

[21] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, "Learning distributed caching strategies in small cell networks," in *Wireless Communications Systems (ISWCS), 2014 11th International Symposium on*. IEEE, 2014, pp. 917–921.

[22] J. Song, M. Sheng, T. Q. Quek, C. Xu, and X. Wang, "Learning-based content caching and sharing for wireless networks," *IEEE Transactions on Communications*, vol. 65, no. 10, pp. 4309–4324, 2017.

[23] S. O. Somuyiwa, D. Gündüz, and A. György, "Reinforcement learning for proactive caching of contents with different demand probabilities," in *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*, 2018, pp. 1–6.

[24] C. Zhong, M. C. Gursoy, and S. Velipasalar, "Deep reinforcement learning-based edge caching in wireless networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 1, pp. 48–61, 2020.

[25] L. Jiang, G. Feng, and S. Qin, "Content distribution for 5G systems based on distributed cloud service network architecture," *KSII Transactions on Internet and Information Systems*, vol. 9, no. 11, pp. 4268–4290, 2015.

[26] R. Li, Z. Zhao, X. Chen, J. Palicot, and H. Zhang, “TACT: A transfer actor-critic learning framework for energy saving in cellular radio access networks,” *IEEE Transactions on Wireless Communications*, vol. 13, no. 4, pp. 2000–2011, 2014.

[27] H. Kim, D. Veciana, X. Yang, and Venkatachalam, “Alpha-optimal user association and cell load balancing in wireless networks,” in *Computer Communications (INFOCOM), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1–9.

[28] A. Nowé, P. Vrancx, and Y.-M. De Hauwere, “Game theory and multi-agent reinforcement learning,” in *Reinforcement Learning*. Springer, 2012, pp. 441–470.

[29] Q. Sun, C. Zhang, N. Jiang, J. Yu, and L. Xu, “Data-driven nonlinear near-optimal regulation based on multi-dimensional taylor network dynamic programming,” *IEEE Access*, vol. 8, pp. 36 476–36 484, 2020.

[30] Y. Wei, F. R. Yu, M. Song, and Z. Han, “User scheduling and resource allocation in HetNets with hybrid energy supply: An actor-critic reinforcement learning approach,” *IEEE Transactions on Wireless Communications*, vol. 17, no. 1, pp. 680–692, 2018.

[31] K. Son, H. Kim, Y. Yi, and B. Krishnamachari, “Base station operation and user association mechanisms for energy-delay tradeoffs in green cellular networks,” *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 8, pp. 1525–1536, 2011.

[32] R. S. Sutton and A. G. Barto, “Reinforcement learning: An introduction,” *MIT press, Cambridge*, 1998.

[33] A. Tavakoli, F. Pardo, and P. Kormushev, “Action branching architectures for deep reinforcement learning,” *arXiv preprint arXiv:1711.08946*, 2017.

[34] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, “A survey of actor-critic reinforcement learning: Standard and natural policy gradients,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[35] W. Jiang, G. Feng, S. Qin, and Y. Liu, “Multi-agent reinforcement learning based cooperative content caching for mobile edge networks,” *IEEE Access*, vol. 7, pp. 61 856–61 867, 2019.

[36] D. Lee, J. Choi, J.-H. Kim, S. H. Noh, S. L. Min, Y. Cho, and C. S. Kim, “LRFU: a spectrum of policies that subsumes the least recently used and least frequently used policies,” *IEEE Transactions on Computers*, vol. 50, no. 12, pp. 1352–1361, 2001.

[37] K. Psounis and B. Prabhakar, “A randomized web-cache replacement scheme,” in *Computer Communications (INFOCOM), 2001 IEEE International Conference on*, vol. 3. IEEE, 2001, pp. 1407–1415.

[38] J. Kwak, Y. Kim, L. B. Le, and S. Chong, “Hybrid content caching in 5G wireless networks: Cloud versus edge caching,” *IEEE Transactions on Wireless Communications*, vol. 17, no. 5, pp. 3030–3045, 2018.

[39] Y. M. Saputra, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, “A novel mobile edge network architecture with joint caching-delivering and horizontal cooperation,” *IEEE Transactions on Mobile Computing*, vol. 20, no. 1, pp. 19–31, 2021.

[40] P. Lin, Q. Song, J. Song, A. Jamalipour, and F. R. Yu, “Cooperative caching and transmission in CoMP-integrated cellular networks using reinforcement learning,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5508–5520, 2020.

[41] Z. Ning, P. Dong, X. Kong, and F. Xia, “A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things,” *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4804–4814, 2019.

[42] K. Poularakis, G. Iosifidis, L. Tassiulas *et al.*, “Approximation algorithms for mobile data caching in small cell networks,” *IEEE Transactions on Communications*, vol. 62, no. 10, pp. 3665–3677, 2014.

[43] F. M. Harper and J. A. Konstan, “The movielens datasets: History and context,” *ACM Transactions on Interactive Intelligent Systems (TIIS)*, vol. 5, no. 4, p. 19, 2016.

[44] http://grouplens.org/datasets/movielens/100k/.

[45] S. Muller, O. Atan, M. van der Schaar, and A. Klein, “Context-aware proactive content caching with service differentiation in wireless networks,” *IEEE Transactions on Wireless Communications*, vol. 16, no. 2, pp. 1024–1036, 2017.

Wei Jiang received the B.S. degree in School of Communication and Information Engineering at Chongqing University of Posts and Telecommunications (CQUPT) in 2013, and the Ph.D. degree in School of Communication and Information Engineering at University of Electronic Science and Technology of China (UESTC) in 2019. She was a visiting Ph.D. student at the Pennsylvania State University (PSU) from 2017 to 2018. She is currently a post-doctor with the College of Electronics and Information Engineering, Shenzhen University. Her current research interests include next generation mobile communication systems, machine learning and content caching and distribution techniques in mobile networks.

Daquan Feng received the Ph.D. degree in information engineering from the National Key Laboratory of Science and Technology on Communications, University of Electronic Science and Technology of China, Chengdu, China, in 2015. From 2011 to 2014, he was a visiting student with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA. After graduation, he was a Research Staff with State Radio Monitoring Center, Beijing, China, and then a Postdoctoral Research Fellow with the Singapore University of Technology and Design, Singapore. Since 2016, he has been with the College of Electronics and Information Engineering, Shenzhen University, as an Assistant Professor and then Associate Professor. His research interests include URLLC communications, MEC, and massive IoT networks. Dr. Feng is an Associate Editor of IEEE COMMUNICATIONS LETTERS.

Yao Sun is currently a Lecturer with James Watt School of Engineering, the University of Glasgow, Glasgow, UK. Dr. Sun has extensive research experience in wireless communication area. He has won the IEEE Communication Society of TAOS Best Paper Award in 2019 ICC. He has been the guest editor for special issues of several international journals. He has been served as TPC Chair for UCET 2021, and TPC member for number of international conferences, including VTC spring 2022, GLOBECOM 2020, WCNC 2019, ICCT 2019. His research interests include intelligent wireless networking, network slicing, blockchain system, internet of things and resource management in mobile networks. Dr. Sun is a senior member of IEEE.

Gang Feng (M’01, SM’06) received his BEng and MEng degrees in Electronic Engineering from the University of Electronic Science and Technology of China (UESTC), in 1986 and 1989, respectively, and the Ph.D. degrees in Information Engineering from The Chinese University of Hong Kong in 1998. He joined the School of Electric and Electronic Engineering, Nanyang Technological University in December 2000 as an assistant professor and was promoted as an associate professor in October 2005. At present he is a professor with the National Laboratory of Communications, University of Electronic Science and Technology of China.

Dr. Feng has extensive research experience and has published widely in computer networking and wireless networking research. His research interests include AI enabled wireless access and content distribution, next generation cellular networks, etc. Dr. Feng is a senior member of IEEE.

Zhenzhong Wang received the Ph.D. degree in electromagnetic field and microwave technology from Beijing University of Posts and Telecommunications, Beijing, China, in 2010. Since 2010, he has been with the Technology and Management Center of China Central Television which is renamed as China Media Group in 2018. Now, Dr. Wang is a Professorate Senior Engineer. His research interests include 4K/8K UHD production, media contents distribution and 5G transmission.

Xiang-Gen Xia (M'97, S'00, F'09) received his B.S. degree in mathematics from Nanjing Normal University, Nanjing, China, and his M.S. degree in mathematics from Nankai University, Tianjin, China, and his Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, in 1983, 1986, and 1992, respectively.

He was a Senior/Research Staff Member at Hughes Research Laboratories, Malibu, California, during 1995-1996. In September 1996, he joined the Department of Electrical and Computer Engineering, University of Delaware, Newark, Delaware, where he is the Charles Black Evans Professor. His current research interests include space-time coding, MIMO and OFDM systems, digital signal processing, and SAR and ISAR imaging. Dr. Xia is the author of the book Modulated Coding for Intersymbol Interference Channels (New York, Marcel Dekker, 2000).

Dr. Xia received the National Science Foundation (NSF) Faculty Early Career Development (CAREER) Program Award in 1997, the Office of Naval Research (ONR) Young Investigator Award in 1998, and the Outstanding Overseas Young Investigator Award from the National Nature Science Foundation of China in 2001. He received the 2019 Information Theory Outstanding Overseas Chinese Scientist Award, The Information Theory Society of Chinese Institute of Electronics. Dr. Xia has served as an Associate Editor for numerous international journals including IEEE Transactions on Signal Processing, IEEE Transactions on Wireless Communications, IEEE Transactions on Mobile Computing, and IEEE Transactions on Vehicular Technology. Dr. Xia is Technical Program Chair of the Signal Processing Symp., Globecom 2007 in Washington D.C. and the General Co-Chair of ICASSP 2005 in Philadelphia.