IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 18, 2019, accepted May 8, 2019, date of publication May 13, 2019, date of current version May 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2916314

Multi-Agent Reinforcement Learning Based Cooperative Content Caching for Mobile Edge Networks

WEI JIANG, GANG FENG, (Senior Member, IEEE), SHUANG QIN, (Member, IEEE), AND YIJING LIU

National Key Laboratory of Science and Technology on Communications, University of Electronic Science and Technology of China, Chengdu 611731, China

Corresponding author: Gang Feng (fenggang@uestc.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61631004 and Grant 61871099.

ABSTRACT To address the drastic growth of data traffic dominated by streaming of video-on-demand files, mobile edge caching/computing (MEC) can be exploited to develop intelligent content caching at mobile network edges to alleviate redundant traffic and improve content delivery efficiency. Under the MEC architecture, content providers (CPs) can deploy popular video files at MEC servers to improve users' quality of experience (QoE). Designing an efficient content caching policy is crucial for CPs due to the content dynamics, unknown spatial-temporal traffic demands, and limited service capacity. The knowledge of users' preference is very useful and important for efficient content caching, yet often unavailable in advance. Under this circumstance, machine learning can be used to learn the users' preference based on historical demand information and decide the video files to be cached at the MEC servers. In this paper, we propose a multi-agent reinforcement learning (MARL)-based cooperative content caching policy for the MEC architecture when the users' preference is unknown and only the historical content demands can be observed. We formulate the cooperative content caching problem as a multi-agent multi-armed bandit problem and propose a MARL-based algorithm to solve the problem. The simulation experiments are conducted based on a real dataset from MovieLens and the numerical results show that the proposed MARL-based cooperative content caching scheme can significantly reduce content downloading latency and improve content cache hit rate when compared with other popular caching schemes.

INDEX TERMS Caching, cooperative, mobile edge caching, multi-agent reinforcement learning.

I. INTRODUCTION

With the development of smart mobile devices (*e.g.*, smartphones, tablets, wearable devices), the demand for low-latency mobile applications as well as multimedia services has been increasing greatly. It will lead to an explosive mobile traffic which challenges the design of future mobile networks. According to the research from Cisco [1], the total volume of mobile data traffic will rise sevenfold from 2016 to 2021. Studies [2], [3] revealed that a large amount of the raised mobile data traffic is due to the duplicate downloads of popular video files. Using mobile content distribution techniques, popular video files can be cached in intermediate servers or proxies (*e.g.*, gateways, routers) so that the requests for the same video file can be available without duplicate transmissions from remote servers. In this way, the transmission resource consumption at both backhaul and core networks can be significantly saved and also the Quality of Experience (QoE) of users could be improved [4].

Content distribution networks (CDN) have been well investigated in the Internet [5]. However, we cannot directly apply traditional CDN based content caching techniques to mobile networks, since content caching mechanism in mobile networks is quite different from that in the Internet. In mobile networks, the network resources (*e.g.*, computation capacity, bandwidth capacity, storage capacity) and the locations of the deployed servers are constrained. Furthermore, the hit rate of cached video files could be rather low in mobile networks due to the user mobility, content dynamics and limited number of users in a cell. Moreover, the amount of video files provided

The associate editor coordinating the review of this manuscript and approving it for publication was Zhiyong Chen.

by content providers (CPs) is increasing rapidly and the updating rate becomes increasingly high. Although caching cost is becoming much cheaper, it is impossible to cache all video files as caching less-popular video files at servers still needs to consume storage and backhaul resources. Hence, it is crucial to design efficient caching policies to maximize the benefits of local caching and sharing for future mobile networks.

Recently, the emerging mobile edge caching/computing (MEC) architectural technology has been standardized by the European Telecommunications Standards Institute [6]. MEC servers provide high caching and computing capabilities within the mobile network edges, which can be exploited to deploy mobile applications and multimedia services as well as to cache popular video files in close proximity to mobile subscribers [7]. Furthermore, MEC servers can provide specific functions that cannot be fulfilled with traditional network infrastructure, such as mobile big data analytics, context-aware services performance optimization. Moreover, MEC servers in a cluster can perform caching in a cooperative way and share their cached video files with each other [8]. MEC servers could be owned by CPs aiming at improving their users' QoE. Under this circumstance, accurate cell information (such as users' demands, users' context information, radio conditions, *etc.*) can be acquired dynamically to facilitate intelligent content caching in a cooperative way to improve cache hit rate and alleviate user perceived downloading latency.

The main challenge of efficient content caching concerns about content popularity which is defined as the probability that a specific file is requested at a certain time. Most existing researches on proactive caching assume the content popularity distribution is known or obeys Zipf distribution or its variants [9]–[15]. In reality, the content popularity is complex and non-stationary due to the content dynamics and unknown spatial-temporal traffic demands. In this case, learning methods could be used to predict content popularity from traffic and content access pattern [16]–[20]. However, content popularity can well reflect the average interests of multiple users, but may not reflect the interests of individual users. In fact, the users' preference in terms of interested files may substantially differ from each other [21]. Hence, an efficient proactive caching policy should take into account both the users' preference and the CP's specific objective. In particular, many CPs have provided service differentiation to their users, *e.g.*, members have more benefits than non-members. CPs may be willing to optimize content caching according to users' prioritization level. For example, if there are users with different interests, the CP would like to prioritize members by caching the video files favored by members. Furthermore, if a CP wants to promote a certain video file, the CP could prioritize the video file in content caching strategies [22].

In this paper, we design a cooperative content caching scheme in MEC servers by exploiting multi-agent reinforcement learning when the prior information of content popularity and users' preference are unknown. Considering service differentiation, we use the weighted reduction of downloading latency as the caching reward and model the cooperative content caching problem as a multi-agent multi-armed bandit problem aiming at maximizing the accumulated expected caching reward over a long-term horizon. Q-learning is used by MEC servers to learn how to coordinate their caching decisions in the multi-agent system. MEC servers learn the Q-values of their own caching decisions in conjunction with those of other MEC servers. Since the space of Q-table is huge and thus traditional multi-agent Q-learning algorithm may need exponential number of steps to traverse all the Q-values, we propose a combinatorial upper confidence bound method to reduce the Q-table space to effectively reduce the complexity. Simulation experiments are conducted based on a real dataset from MovieLens and the numerical results show that our proposed caching scheme can significantly reduce content downloading latency and improve content cache hit rate when compared with other popular caching schemes.

The remainder of the paper is organized as follows. We introduce previous related work and describe the system model in Section II and Section III respectively. In Section IV, the optimal cooperative content caching problem is formulated as a multi-agent multi-armed bandit problem. In Section V, we elaborate the multi-agent reinforcement learning based algorithm to solve the problem. In Section VI, we evaluate the performance of our proposed caching algorithm by simulation and finally conclude the paper in Section VII.

II. RELATED WORK

In recent years, the design of efficient content caching strategies for mobile networks has attracted much research interest. Existing mobile caching strategies can be roughly categorized into two types: those with perfect content popularity information and those without.

A. CACHING STRATEGIES WITH PERFECT POPULARITY INFORMATION

So far, the majority of related research work assumes that the perfect popularity information is known in advance. In [9], the authors investigated the fundamental limits of caching in a caching system. The authors of [10] proposed coordinated caching at base-stations (BSs) to alleviate backhaul transmission as well as improve the users' QoE. In [11], the authors studied a caching problem in a heterogeneous network with helpers. The caching problem was shown to be NP-hard, and a heuristic algorithm was proposed. In [12], wireless bandwidth constraints are set in the content caching problem of small BSs (SBSs). In [13], a joint design of caching and delivery strategies was studied by assuming the users' instantaneous content demands are known in advance. The authors of [14] presented a novel architecture for Device-to-Device (D2D) caching with cooperation, and formulated the D2D caching problem aiming at minimizing the downloading latency. In [15], we focus on hybrid SBS-D2D caching and propose an optimal cooperative caching policy

with aim of providing a global optimal caching solution for hybrid SBS-D2D networks. These studies assumed the perfect knowledge of instantaneous content demands or the content popularity distribution. However, due to the content dynamics and users' mobility, the content popularity in a cell may change over time. Therefore, it is a necessary and challenging work to learn the content popularity and users' preference dynamically.

B. CACHING STRATEGIES WITH NON-PERFECT POPULARITY INFORMATION

Thus far, there is little research work which assumes non-perfect information of content popularity, and machine learning methods are exploited to learn the content popularity by observing users' historical content demands and user-content correlations. In [16] and [17], the authors proposed transfer learning-based caching schemes to exploit the contextual information (*i.e.*, users' historical content demands, user-content correlations, social ties, *etc.*) extracted from the source cell. Then, this prior information is introduced to the target cell for finding the optimal content caching strategy. Those transfer learning-based caching schemes exploit supervised learning and thus require training sets. As appropriate training sets are always not at hand, it is more suitable to use an online learning algorithm to estimate the content popularity over time.

In [18], [19] and [20], multi-armed bandit learning is used to online estimate the content popularity. The estimated content popularity is then used to optimize the content caching scheme. The users' connectivity to the SBS and cooperation among SBSs are taken in to consideration respectively in [19] and [20]. However, the estimated content popularity distribution of a large population is very different from the preference of individual users [21]. Thus these approaches are not effective for mobile content caching.

The authors of [21] optimized caching strategy at BSs by exploiting users' preference. They showed that content popularity and users' preference may be different. Hence, content popularity may not reflect the interests of individual users. In [22], the authors proposed a context-aware proactive content caching policy while considering users' preference and service differentiation. The proposed algorithm learns content popularity online leveraging users' context information. However, the authors considered either single cache case [22] or multiple cache case without cooperation [21]. These observations inspire us to develop new efficient caching policy for multiple cache cases with cooperative content caching and delivery.

III. SYSTEM MODEL

MEC brings the cloud computing and caching capabilities to the mobile network edges. Operators can open the MEC servers to CPs, allowing them to deploy video files in the local cache of MEC servers to improve mobile subscriber's experience [23]. We consider a typical MEC system model as shown in Fig. 1. In this model, a set of MEC servers

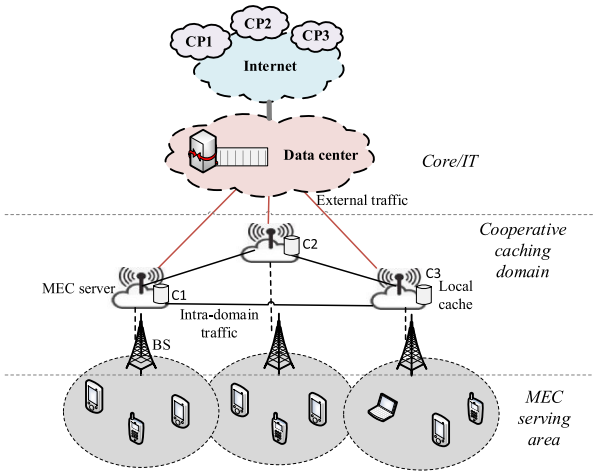

FIGURE 1. System model of cooperative content caching for MEC.

in a cluster form a cooperative caching domain to support mobile multimedia service in the edge of mobile networks. Usually MEC servers could be co-located with the cellular BSs [8] and a MEC server takes charge of providing mobile multimedia services and applications for a cell [17]. For simplicity, in the rest of this paper, we use "domain" and "server" to indicate "cooperative caching domain" and "MEC server" respectively.

A CP owns some resources (such as storage capacity, computation capacity) of servers to provide low-latency local cloud services. The CP periodically deploys and updates popular video files in the local cache of servers to improve mobile users' QoE.

In a specific domain, servers perform content caching and sharing in a cooperative way [8]. For example, if a content request cannot be fulfilled by local cache C1 in the system of Fig. 1, it may be served by C2 or C3. By using the content caching services provided by servers, mobile users' content requests may not need to be forwarded to data center through backhaul links, and thus content delivery latency and redundant transmissions can be drastically reduced.

The servers in a domain can exchange caching information or share cached video files with other servers via the communications between BSs. For example, the X2 interface which provides the data transfer function between NodeBs in LTE [24] can be used to realize communications between these servers. Although cooperative caching is resource-demanding and may cause some latency due to cooperation, it can greatly improve cache hit rate and reduce downloading latency (through the network core) in return. Furthermore, the latency using wireless connection between servers is negligible compared with that for a user to obtain files from core network through backhaul links if the cache hit rate is low without cooperation.

Let there be M servers in a domain, denoted by $\mathcal{M} = \{1, 2, \cdots, M\}$, which are co-located with BSs. We assume that time is slotted into periods. We denote by

$\mathcal{T} = \{1, 2, \cdots, T\}$ the set of time periods, where T is the finite time horizon. Caching decisions are periodically updated for each time period.

There are U users in the domain. Let $\mathcal{U} = \{1, 2, \ldots, U\}$ be the set of considered users. The CP provides the total number of F video files and the set of video files is denoted by $\mathcal{F} = \{1, 2, \cdots, F\}$. Each file $f \in \mathcal{F}$ is of size s_f and the CP owns local caches with storage capacity S_m in each server $m \in \mathcal{M}$.

Each BS has a certain coverage area. Users located in the coverage area can connect to the server co-located with the BS. The set of connected users of a server may change dynamically due to the user mobility. We define $\mathcal{U}_m^t \subset \mathcal{U}$ as the set of users located in the coverage area served by server m in time period t.

Users' preference is the conditional probability that a specific file is requested by a user given that the user has a file request, which reflects the demands of each user. Denote by $p_{f|u} \in [0, 1]$ the conditional probability that user u demands file f given that it requests a file. Users may have different preferences for files. We assume that the content demands of user u occurs independently and follows a Poisson process with mean ϕ_u (arrival/time period) [25].

We denote by $d_{u,f}^t$ the frequency of file f requested by user u in time period t, which is an independent identically distributed (*i.i.d.*) random variable with mean $\theta_{u,f} = E\left(d_{u,f}^t\right)$. Therefore, the expected frequency of file f requested by user u within one time period is $\theta_{u,f} = \phi_u \cdot p_{f|u}$. In this paper, we take a realistic scenario that ϕ_u and $p_{f|u}$ are unknown in advance.

We assume that each server has the information of currently cached files by other servers in the same domain and periodically broadcasts to its connected users. Users send a file request to their connected server, other servers in the domain or data center, which is up to the availability of the requested file [20]. Specifically, if the requested file is unavailable at their connected server, the server just forwards the content request and does not know the detailed information about the requested file. Then, a MEC server can obtain the knowledge of users' instantaneous content demand when the requested file is available in its cache, but it may not know users' instantaneous content demand when the requested file is unavailable in its cache. Therefore, the overall users' instantaneous content demand is unknown to these servers.

When a user has a content demand, he can get the requested file in the following ways: 1) Local transmission: if the local server (*i.e.*, he connected server) has stored the requested file in its cache, and then the requested file is transmitted from the local server to the user directly. 2) Intra-domain transmission: if the file is not stored in the local server, but at least one of servers in the domain have stored it, and then the local server fetches the requested file from other servers for the user. 3) External transmission: if all servers in the domain do not store the requested file, the local server fetches the requested file from the data center.

We denote the downloading rate between server m and server n as $Z_{m,n}$, $n \in \mathcal{M}, n \neq m$, and the downloading rate between server m and data center as $Z_{m,0}$, and $Z_{m,n} > Z_{m,0}$. We assume that the downloading latency of $u \in \mathcal{U}_m^t$ incurred for fetching file f from server m is $L_{u,m,f}^t = 0$. The downloading latency of $u \in \mathcal{U}_m^t$ fetching file f from MEC server n and data center is $L_{u,n,f}^t = s_f / Z_{m,n}$, $n \in \mathcal{M}, n \neq m$ and $L_{u,0,f}^t = s_f / Z_{m,0}$ respectively.

Naturally, CPs should design the content caching strategy carefully to minimize the average downloading latency for maximizing the user QoE. However, the CP may intend to provide different services to users. For example, if there are users with different interests, the CP can prioritize some users by caching files interested by these users. In this case, the downloading latency of a prioritized user can be associated with a higher weight than the downloading latency of a regular user. To this end, we denote by $\mathcal{K}$ the set of service types. Let $v_k \geq 1$ denote a weight associated with the downloading latency experienced by a user of service type $k \in \mathcal{K}$. Considering the service differentiation, the CP's goal becomes to minimize the weighted downloading latency. Furthermore, the CP may want to prioritize certain files, such as movies with high rating. In this case, each file can be associated with a fixed and known weight. Hence, we denote by $w_f \geq 1$ the prioritization weight for file f. In case of no service differentiation, *i.e.*, $v_k = 1$, $k \in \mathcal{K}$ and $w_f = 1, f \in \mathcal{F}$, the goal is to minimize the average (non-weighted) downloading latency. This is a special case of our model. The notations we used in this paper are listed in Table 1.

TABLE 1. Notations.

Symbol	Definition
$\mathcal{M}$	The set of MEC servers
$\mathcal{T}$	The set of time periods
$\mathcal{U}$	The set of users
$\mathcal{U}_m^t$	The set of users located in the coverage area served by server $m \in \mathcal{M}$ in time period $t \in \mathcal{T}$
$\mathcal{F}$	The set of files
s_f	The size of file $f \in \mathcal{F}$
S_m	The storage capacity of MEC server $m \in \mathcal{M}$
$d_{u,f}^t$	The number of times that user $u \in \mathcal{U}$ requests file $f \in \mathcal{F}$ within time period $t \in \mathcal{T}$
$\theta_{u,f}$	The expected number of times that user $u \in \mathcal{U}$ requests file $f \in \mathcal{F}$ within a time period
$Z_{m,n}$	The downloading rate between server $m \in \mathcal{M}$ and server $n \in \mathcal{M}$
$Z_{m,0}$	The downloading rate between server $m \in \mathcal{M}$ and data center
$L_{u,n,f}^t$	The downloading latency of $u \in \mathcal{U}_m^t$ fetching file $f \in \mathcal{F}$ from MEC server $n \in \mathcal{N}$
$L_{u,0,f}^t$	The downloading latency of $u \in \mathcal{U}_m^t$ fetching file $f \in \mathcal{F}$ from data center
$\mathcal{K}$	The set of service types
v_k	The weight associated with the downloading latency experienced by a user of service type $k \in \mathcal{K}$
w_f	The prioritization weight for file $f \in \mathcal{F}$

IV. PROBLEM FORMULATION

In this section, we model the cooperative content caching problem of minimizing the weighted downloading latency as a multi-agent multi-armed bandit (MAB) problem.

A. PRELIMINARY: MULTI-AGENT MULTI-ARMED BANDIT PROBLEM

MAB problem has been well studied in statistics and machine learning for many years. In the classic MAB problem, there are a single agent and F independent arms. In each time period, the agent selects an arm to play, and receives a random reward of the played arm. The expected reward of playing an arm follows an *i.i.d.* model with an unknown mean. The problem is to decide which arm should be played in the next time period based on the current information, so that the accumulated expected reward in a long-term can be maximized.

In a MAB learning problem, there is a well-known tradeoff between exploration and exploitation: whether one should select some new arms that have not been played much, aiming at reliably estimating the mean rewards of these arms (exploration) or one should select known arms that offer higher rewards so far, aiming at achieving the highest accumulated empirical rewards (exploitation). If the agent knows the reward model of each arm, the optimal algorithm would always play the arm with the maximum expected reward. A commonly used criterion for measuring the performance of a MAB algorithm is *regret*, which is defined as the gap between the obtained reward of the algorithm and that of the optimal algorithm. Auer *et al.* [26] proposed an index policy, called upper confidence bound (UCB) algorithm, which can achieve a regret on the order of $O(\log T)$ over time horizon T, and this is the order-optimal.

There is a variant of MAB problem, called the combinatorial MAB (CMAB) [27]. Different from the classic MAB model, in the CMAB model, a set of arms (called a *super arm*) can be played simultaneously at each time period. The combinatorial nature among multiple super arms results in dependencies between them. Further, the number of super arms increases super-exponentially and the reward of each super arm depends on the rewards of all underlying arms. Extended to multi-agent setting, the expected reward of playing a super arm is associated with the play of other agents. In the following, we model the cooperative content caching problem to minimize the weighted downloading latency as a multi-agent CMAB problem.

B. MULTI-AGENT CMAB PROBLEM FOR CONTENT CACHING IN MEC SERVERS

We consider an F-armed bandit with M agents, and each arm and agent corresponds to a file and a server respectively. In each time period, the agents select several arms to play, *i.e.*, servers select several files to cache. The aim is to minimize the weighted downloading latency.

We define a binary caching decision variable $x^t_{m,f} \in \{0,1\}$ to indicate whether file f is placed at the local cache of server m in time period t: $x^t_{m,f} = 1$ if file f is cached and $x^t_{m,f} = 0$ otherwise. The total size of cached files in a server cannot exceed its storage capacity, *i.e.*, $\sum_{f\in\mathcal{F}} x^t_{m,f}\cdot s_f \le S_m, \forall m \in \mathcal{M}$. Then, the caching decision vector of server m in time period t is defined as $\boldsymbol{x}^t_m = (x^t_{m,1},\ldots,x^t_{m,F})^T$, where $x^t_{m,f} \in \{0,1\}, f \in \mathcal{F}$ and $\sum_{f\in\mathcal{F}} x^t_{m,f}\cdot s_f \le S_m$. The caching decision matrix of all servers in time period t is denoted by $\mathbf{X}^t = (\boldsymbol{x}^t_1,\ldots,\boldsymbol{x}^t_M)$.

We define the binary caching routing decision variable $y^t_{u,n,f} \in \{0,1\}$ to indicate whether user $u \in \mathcal{U}^t_m$ retrieves file f from server n or not. Obviously, user u can fetch file f from server n only when file f is placed at the local cache of server n and thus, $y^t_{u,n,f} \le x^t_{n,f}$. Furthermore, file f can be retrieved from either one server in the domain or the data center. Hence, we have $\sum_{n\in\mathcal{M}} y^t_{u,n,f} \le 1$. In our content retrieval protocol, user $u \in \mathcal{U}^t_m$ always retrieves file f from the server with the lowest delay, namely $y^t_{u,n,f} = 1$ when $\prod_{j=1}^{i-1}(1 - x^t_{(j)_u,f})\cdot x^t_{(i)_u,f} = 1$, where $n = (i)_u$ is the i-th lowest delay server for user u and $y^t_{u,n,f} = 0$ otherwise.

We use the weighted reduction of downloading latency as the reward of server m caching file f in time period t, which is defined as:

$$r^t_{m,f} = w_f \cdot \sum_{n\in\mathcal{M}}\sum_{u\in\mathcal{U}^t_n} d^t_{u,f}\cdot v_{k_u}\cdot (L^t_{u,0,f} - L^t_{u,m,f})\cdot y^t_{u,m,f},$$

where $k_u \in \mathcal{K}$ is the service type of user u. The expected reward of server m caching file f is given by:

$$E\left(r^t_{m,f}\right) = w_f \cdot \sum_{n\in\mathcal{M}}\sum_{u\in\mathcal{U}^t_n} \theta_{u,f}\cdot v_{k_u}\cdot (L^t_{u,0,f} - L^t_{u,m,f})\cdot y^t_{u,m,f}.$$

In time period t, for server m, a particular caching decision vector $\boldsymbol{x}^t_m$ is selected and only for those files with $x^t_{m,f} = 1$, the value of $r^t_{m,f}$ is observed. We denote by $\mathcal{C}^t_m = \{f \mid f \in \mathcal{F}, x^t_{m,f} = 1\}$ the set of files cached in server m in time period t. In time period t, the history seen by server m is:

$$\mathcal{H}^t_m = \{(\mathbf{X}^1 \to (r^1_{m,f})_{f\in\mathcal{C}^1_m}), \ldots, (\mathbf{X}^t \to (r^t_{m,f})_{f\in\mathcal{C}^t_m})\},$$

which implies that each server can observe the caching decisions of other servers.

The problem we need to solve can be presented as follows. Given the network topology and MEC server storage capacity, without knowing the users' content demand in advance, by simply observing the content requests for the cached files over time, what is the optimal caching strategy such that the total expected caching reward can be maximized?

The proactive caching problem with objective of maximizing the total expected caching reward up to time horizon T is formulated as follows:

$$\begin{aligned}
&\max \sum_{t=1}^{T}\sum_{m\in\mathcal{M}}\sum_{f\in\mathcal{F}} E\left(r^t_{m,f}\right),\\
&s.t. \sum_{f\in\mathcal{F}} x^t_{m,f}\cdot s_f \le S_m, \quad \forall m \in \mathcal{M},\\
&\quad x^t_{m,f} \in \{0,1\}, \forall m \in \mathcal{M}, \quad \forall f \in \mathcal{F}. \qquad (1)
\end{aligned}$$

It is easy to show that even with the knowledge of the expected users' content demand $\theta_{u,f}, u \in \mathcal{U}, f \in \mathcal{F}$ at each server, problem (1) is NP-hard [17]. Moreover, we devote to a more practical case in which $\theta_{u,f}, u \in \mathcal{U}, f \in \mathcal{F}$ is unknown in advance. We then resort to multi-agent reinforcement learning process, in which servers learn from the historical caching reward to adjust their caching decision, to solve problem (1).

V. MULTI-AGENT REINFORCEMENT LEARNING-BASED CACHING POLICY

Extending reinforcement learning to multi-agent systems has been an attractive research area in recent years. A well-understood example of reinforcement learning algorithm is Q-learning. Due to the simplicity and convergence guarantees of Q-learning, it is a nature choice to apply Q-learning algorithm to multi-agent systems. In multi-agent Q-learning algorithm, each agent maintains Q-values as the expected rewards for every state-action pairs, which are stored in an array known as the Q-table. Since the multi-agent CMAB problem does not require a state representation, we just simplify the general multi-agent Q-learning to its stateless version [28].

We denote the set of caching decision vectors by $\mathcal{X}_m = \{(x_{m,1}, \ldots, x_{m,F})^T \,|x_{m,f} \in \{0,1\}, f \in \mathcal{F}, \sum_{f\in\mathcal{F}} x_{m,f} \cdot s_f \leq S_m\}$ and the set of reduced caching decision matrixes by $\mathcal{X}_{-m} = \{(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{m-1}, \boldsymbol{x}_{m+1}, \ldots, \boldsymbol{x}_M) \,|\boldsymbol{x}_n \in \mathcal{X}_n, n \in \mathcal{M}, n \neq m\}$ for server m. In classical multi-agent Q-learning algorithm, each server $m \in \mathcal{M}$ maintains a Q-table $\{Q_{m,\boldsymbol{x}_m}(\boldsymbol{X}_{-m}) | \boldsymbol{x}_m \in \mathcal{X}_m, \boldsymbol{X}_{-m} \in \mathcal{X}_{-m}\}$, where Q-value $Q_{m,\boldsymbol{x}_m}(\boldsymbol{X}_{-m})$ provides the estimated reward of selecting caching decision vector $\boldsymbol{x}_m$ when the reduced caching decision matrix is $\boldsymbol{X}_{-m}$. Server m updates its Q-table based on sample $\mathbf{X}^t \rightarrow (r^t_{m,f})_{f\in\mathcal{C}^t_m}$ as follows: $Q_{m,\boldsymbol{x}^t_m}(\boldsymbol{X}^t_{-m}) \leftarrow Q_{m,\boldsymbol{x}^t_m}(\boldsymbol{X}^t_{-m}) + \lambda_t \cdot (\sum_{f\in\mathcal{C}^t_m} r^t_{m,f} - Q_{m,\boldsymbol{x}^t_m}(\boldsymbol{X}^t_{-m}))$, where $0 \leq \lambda_t \leq 1$ is the learning rate, which usually decays with t. $\boldsymbol{x}^t_m$ and $\boldsymbol{X}^t_{-m}$ are the caching decision vector selected by server m and the reduced caching decision matrix observed by server m in time period t respectively.

However, there are two issues preventing the use of the classical multi-agent Q-learning algorithm to solve the formulated problem. First, the space of the Q-table is 2^{M+F}. It is exponential to the number of servers and the number of files, and thus the classical multi-agent Q-learning algorithm may need exponential number of steps to traverse all the values of the Q-table. Second, after caching decision vector $\boldsymbol{x}_m$ is selected, we can get some information regarding the rewards of the underlying caching decision variables $x_{m,f}, f \in \mathcal{F}$, which may be shared by other caching decision vectors. However, these information are discarded in the classical multi-agent Q-learning algorithm, making it less effective.

In the CMAB model, a caching decision vector (*e.g.*, $\boldsymbol{x}_m$) consists of underlying caching decision variables (*e.g.*, $x_{m,f}$, $f \in \mathcal{F}$). In each time period, after one caching decision vector is selected, the rewards of all underlying caching decision variables are revealed. Since the expected rewards of caching decision vectors can be computed by the expected rewards of underlying caching decision variables, a combinatorial UCB (CUCB) algorithm [27] is proposed to use the expected rewards of underlying caching decision variables instead of the expected rewards of caching decision vectors. Thus, the space of the Q-table can be reduced from the number of caching decision vectors to the number of underlying caching decision variables.

Therefore, we introduce a CUCB-type algorithm into our multi-agent system to reduce the space of the Q-table. We denote the set of reduced caching decision vectors by $\mathcal{X}_{-m,f} = \{(x_{1,f}, \ldots, x_{m-1,f}, x_{m+1,f}, \ldots, x_{M,f}) \,|x_{n,f} \in \{0,1\}, n \in \mathcal{M}, n \neq m\}$ for server m. We employ Q-table $\{Q_{m,f}(\boldsymbol{x}_{-m,f}) \mid f \in \mathcal{F}, \boldsymbol{x}_{-m,f} \in \mathcal{X}_{-m,f}\}$ instead of $\{Q_{m,\boldsymbol{x}_m}(\boldsymbol{X}_{-m}) \mid \boldsymbol{x}_m \in \mathcal{X}_m, \boldsymbol{X}_{-m} \in \mathcal{X}_{-m}\}$ to reduce the space of Q-table from 2^{M+F} to $F \cdot 2^{M-1}$ for each server $m \in \mathcal{M}$. Instead, this reduction is at the cost of computing the optimal caching decision vector; *i.e.*, the algorithm has to employ a computation oracle which takes the Q-table as input, together with the problem instance, for computing the optimal caching decision vector $\boldsymbol{x}^*_m$. As the optimal solution of a combinatorial problem may be computationally hard, we allow the oracle to be an (α, β)-approximation oracle, where $\alpha, \beta \leq 1$, *i.e.*, the oracle could output a caching decision vector whose expected reward is at least α fraction of the optimal expected reward with probability β. Next, we elaborate our proposed multi-agent reinforcement learning based caching algorithm as follows.

Q-table $\{Q_{m,f}(\boldsymbol{x}_{-m,f}) \mid f \in \mathcal{F}, \boldsymbol{x}_{-m,f} \in \mathcal{X}_{-m,f}\}$: Q-value $Q_{m,f}(\boldsymbol{x}_{-m,f})$ is the average reward observed by server m of caching file f when the reduced caching decision vector is $\boldsymbol{x}_{-m,f}$.

Updating Q-values: If file f is cached by server m in time period t, Q-table is updated as $Q_{m,f}(\boldsymbol{x}^t_{-m,f}) \leftarrow Q_{m,f}(\boldsymbol{x}^t_{-m,f}) + \frac{1}{C_{m,f}(\boldsymbol{x}^t_{-m,f})+1} \cdot (r^t_{m,f} - Q_{m,f}(\boldsymbol{x}^t_{-m,f}))$, where $\boldsymbol{x}^t_{-m,f}$ is the reduced caching decision vector observed by server m in time period t and $C_{m,f}(\boldsymbol{x}^t_{-m,f})$ is the number of times that $\boldsymbol{x}^t_{-m,f}$ is observed by server m until time period t.

Belief-maintenance procedure: Although server m currently has Q-values of all reduced caching decision vectors, the expected reward of performing a caching decision vector depends on other servers' current strategies. To estimate other servers' current strategies, each server observes the historical caching decisions of other servers and maintains beliefs about other servers' strategies. The belief-maintenance procedure is presented as follows. Each server $m \in \mathcal{M}$ assumes that server $n \in \mathcal{M}, n \neq m$ will cache files in accordance with m's current beliefs about n (*i.e.*, m's empirical probability distribution over n's caching decisions). Server m treats the relative frequency of server n's caching choices as the indicator of n's current strategy. Server m keeps counts $C_{m,f}$ denoting the number of times file f cached by server m and $C_{m,n,f}$ denoting the number of times that file f cached by server n. For each server $n \neq m$, m assumes n caches file f with probability $\Pr_{m,n,f} = C_{m,n,f}/t$. Hence, server m assesses the probability

of reduced caching decision vector $\boldsymbol{x}_{-m,f} \in \mathcal{X}_{-m,f}$ selected by other servers to be:

$$\Pr_m(\boldsymbol{x}_{-m,f}) = \prod_{\substack{n\in\mathcal{M}\\ n\neq m\\ x_{n,f}=1}} \Pr_{m,n,f} \cdot \prod_{\substack{n\in\mathcal{M}\\ n\neq m\\ x_{n,f}=0}} (1-\Pr_{m,n,f}),$$

and the expected reward of caching file f to be $Q_{m,f} = \sum_{\boldsymbol{x}_{-m,f}\in\mathcal{X}_{-m,f}} Q_{m,f}(\boldsymbol{x}_{-m,f}) \cdot \Pr_{m,f}(\boldsymbol{x}_{-m,f})$.

Biasing Q-values: To promote the exploration-exploitation, we bias the Q-values based on the particular structure of the problem. The biased Q-value is given by $\bar{Q}_{m,f} = Q_{m,f} + l \cdot \sqrt{\frac{3\log(M\cdot t)}{2M\cdot C_{m,f}}}$, where $l = \max_{i\in\mathcal{F}} Q_{m,i}/s_i$.

(α, β)-approximation oracle: We resort to a (α, β) approximation oracle which takes the biased Q-value $\bar{Q}_{m,f}, m \in \mathcal{M}, f \in \mathcal{F}$ as input for finding the optimal caching decision vector $\boldsymbol{x}_m^*$. The problem for computing $\boldsymbol{x}_m^*$ is a 0-1 Knapsack problem with values $\bar{Q}_{m,f}, f \in \mathcal{F}$, and weights $s_f, f \in \mathcal{F}$, which can be rewritten as follows:

$$\begin{aligned} &\max \sum\nolimits_{f\in\mathcal{F}} \bar{Q}_{m,f} \cdot x_{m,f}, \\ &s.t. \sum\nolimits_{f\in\mathcal{F}} s_f \cdot x_{m,f} \le S_m, \\ &\quad x_{m,f} \in \{0,1\}. \end{aligned} \tag{2}$$

The 0-1 Knapsack problem is known to be NP-hard [29]. The exact solution of $\boldsymbol{x}_m^*$ requires huge computational resources. Therefore, we employ a greedy algorithm with low-complexity as the (α, β)-approximation oracle. The solution of the (α, β)-approximation oracle is defined as $\boldsymbol{x}_m = Oracle(\bar{Q}_{m,1}, \ldots, \bar{Q}_{m,F})$. The greedy algorithm starts with the feasible solution $\boldsymbol{x}_m = (0, 0, \ldots, 0)^T$ and sequentially replaces the zeros by ones, starting with the most beneficial (in the sense of values $\bar{Q}_{m,f}/s_f$) if each such change does not break the feasibility. The process is terminated when the last feasible solution is obtained. This is to say that the greedy algorithm constructs a series of feasible solutions with monotonically increasing the objective function value and the last feasible solution is the greedy solution $\boldsymbol{x}_m$. Formally, the solution $\boldsymbol{x}_m$ is obtained in the following way. We sort $x_{m,f_i}, f_i \in \mathcal{F}, i = 1, 2, \ldots, F$ in a descending order according to their "specific values" $\bar{Q}_{m,f_i}/s_{f_i}$: $\bar{Q}_{m,f_1}/s_{f_1} \ge \bar{Q}_{m,f_2}/s_{f_2} \ge \cdots \ge \bar{Q}_{m,f_F}/s_{f_F}$. Let $x_{m,f_1} = 1$ and for $k = 2, \ldots, F$,

$$x^G_{m,f_k} = \begin{cases} 1, & \sum_{j=1}^{k-1} s_{f_j} x^G_{m,f_j} + s_{f_k} \le S_m \\ 0, & \sum_{j=1}^{k-1} s_{f_j} x^G_{m,f_j} + s_{f_k} > S_m. \end{cases}$$

Let $\delta = \sum_{f\in\mathcal{F}} \bar{Q}_{m,f} \cdot x^*_{m,f} / \sum_{f\in\mathcal{F}} \bar{Q}_{m,f} \cdot x_{m,f}$ be the ratio between the objective function value of the optimal algorithm and that of the greedy algorithm. In [29], it has been proved that $\delta \le 2$, and thus $\sum_{f\in\mathcal{F}} \bar{Q}_{m,f} \cdot x_{m,f} \ge \sum_{f\in\mathcal{F}} \bar{Q}_{m,f} \cdot x^*_{m,f}/2$, *i.e.*, the greedy algorithm can achieve at least 1/2 fraction of the optimal objective function value with probability 1. Therefore, the greedy algorithm can be an (α, β)-approximation oracle, where $\alpha = 1/2$ and $\beta = 1$. Additionally, if the maximum file size is much smaller than the storage capacity, then $\delta \approx 1$ [29], and thus, $\alpha \approx 1$. In the case of $s_i = s_j, i, j \in \mathcal{F}$, *i.e.*, when all file sizes are the same, the greedy algorithm is the optimal. The multi-agent reinforcement learning based caching algorithm for MEC is summarized in Algorithm 1.

Algorithm 1 Multi-Agent Reinforcement Learning Based Caching Algorithm

1: **Each server** $m \in \mathcal{M}$ **do**
2: **Initialize:** $C_{m,f} = 0$, $C_{m,n,f} = 0$, $C_{m,f}(\boldsymbol{x}_{-m,f}) = 0$, $Q_{m,f}(\boldsymbol{x}_{-m,f}) = 0, n \in \mathcal{M}\, n \neq m, f \in \mathcal{F}, \boldsymbol{x}_{-m,f} \in \mathcal{X}_{-m,f}$.
3: Cache each file at least once, observe the rewards $r^t_{m,f}, f \in \mathcal{F}$, and update $C_{m,f}$, $C_{m,n,f}$, $C_{m,f}(\boldsymbol{x}_{-m,f})$, $Q_{m,f}(\boldsymbol{x}_{-m,f})$ for $f \in \mathcal{F}, n \in \mathcal{M}, n \neq m, \boldsymbol{x}_{-m,f} \in \mathcal{X}_{-m,f}$.
4: **While** $t \in \mathcal{T}$ **do**
5: Compute $\Pr_{m,n,f} = C_{m,n,f}/t, n \in \mathcal{M}, n \neq m, f \in \mathcal{F}$ and $\Pr_m(\boldsymbol{x}_{-m,f}) = \prod_{\substack{n\in\mathcal{M}\\ n\neq m\\ x_{n,f}=1}} \Pr_{m,n,f} \cdot \prod_{\substack{n\in\mathcal{M}\\ n\neq m\\ x_{n,f}=0}} (1-\Pr_{m,n,f})$, $\boldsymbol{x}_{-m,f} \in \mathcal{X}_{-m,f}, f \in \mathcal{F}$.
6: Compute $Q_{m,f} = \sum_{\boldsymbol{x}_{-m,f}\in\mathcal{X}_{-m,f}} Q_{m,f}(\boldsymbol{x}_{-m,f}) \cdot \Pr_m(\boldsymbol{x}_{-m,f})$, $f \in \mathcal{F}$.
7: Compute $\bar{Q}_{m,f} = Q_{m,f} + l \cdot \sqrt{\frac{3\log(M\cdot t)}{2\cdot M\cdot C_{m,f}}}, f \in \mathcal{F}$.
8: Compute $\boldsymbol{x}^t_m = Oracle(\bar{Q}_{m,1}, \ldots, \bar{Q}_{m,F})$.
9: Broadcast caching decision vector $\boldsymbol{x}^t_m$ to other servers.
10: Receive caching decision vector $\boldsymbol{x}^t_n, n \in \mathcal{M}, n \neq m$ from other servers.
11: Observe $r^t_{m,f}, f \in \mathcal{C}^t_m$.
12: Update $C_{m,f} \leftarrow C_{m,f} + 1, f \in \mathcal{C}^t_m$ and $C_{m,n,f} \leftarrow C_{m,n,f} + 1, n \in \mathcal{M}, n \neq m, f \in \mathcal{C}^t_n$.
13: Update $Q_{m,f}(\boldsymbol{x}^t_{-m,f}) \leftarrow Q_{m,f}(\boldsymbol{x}^t_{-m,f}) + \frac{1}{C_{m,f}(\boldsymbol{x}^t_{-m,f})+1} \cdot (r^t_{m,f} - Q_{m,f}(\boldsymbol{x}^t_{-m,f}))$, $f \in \mathcal{C}^t_m$ and $C_{m,f}(\boldsymbol{x}^t_{-m,f}) \leftarrow C_{m,f}(\boldsymbol{x}^t_{-m,f}) + 1, f \in \mathcal{C}^t_m$.
14: $t \leftarrow t + 1$.

Next, we analyze the complexity of Algorithm 1. In the procedure of Algorithm 1, in each time period, MEC servers make caching decisions in parallel. When a server determines its caching decision, it needs to compute Q-values and find the optimal caching decision by the (α, β)-approximation oracle. The time complexity of computing Q-values is $O(2^{M-1} \cdot F)$ and the time complexity of finding the optimal caching decision vector is $O(F)$. Therefore, the time complexity of Algorithm 1 is $O(2^{M-1} \cdot F)$ for each server in each time period.

In the following, we present the regret bound of Algorithm 1. With an (α, β)-approximation oracle, it is unfair to use the difference between the reward obtained by a CUCB algorithm and the optimal reward as the regret. Instead, we use the (α, β)-approximation regret, which is defined as the difference between the reward obtained by the algorithm and the $\alpha \cdot \beta$ fraction of the optimal reward, since the reward

obtained by the oracle is only α fraction of the optimal reward with probability β.

Fact 1. *The CUCB algorithm using an* (α, β)*-approximation oracle with a monotonicity and bounded smoothness function can achieve an* (α, β)*-approximation regret of* $O(\log T)$ *over time horizon* T.

Proof: cf. [27] for proofs. ■

Corollary 1: Algorithm 1 can achieve a (α, β)*-approximation regret of* $O(\log T)$ *over time horizon* T.

Proof: For server m, the expected reward of a caching decision vector $\boldsymbol{x}_m$ is $\sum_{f \in \mathcal{C}_m^t} r_{m,f}^t$, which is linear to the expected rewards of all underlying caching decision variables $x_{m,f}, f \in \mathcal{F}$. Hence, the monotonicity property is satisfied. We define the bounded smoothness function as $f(\Delta) = S_m \cdot \Delta$, *i.e.*, increasing the expected reward of all underlying caching decision variables by Δ can increase the expected reward of the caching decision vector at most $S_m \cdot \Delta$. According to Fact 1, Algorithm 1 achieves an (α, β)-approximation regret of $O(\log T)$ over time horizon T. ■

VI. PERFORMANCE EVALUATION

We use a dataset of MovieLens from the real word [30] to evaluate the performance of our proposed cooperative content caching algorithm. The MovieLens datasets are full of data describing how people rate movies. The MovieLens 1M Dataset [31] records 6,040 MovieLens users' rating data in the form of ⟨User ID, Movie ID, Rating, Timestamp⟩, which contains 1,000,209 ratings of 3,952 movies within the years 2000 to 2003. Additionally, the users' demographic information is provided in the following format: ⟨User ID, Gender, Age, Occupation, Zip-code⟩.

We assume that the movie rating process in the dataset is corresponding to the content request process (refer to [22] for a similar approach). Therefore, in our simulation, a user rating a movie at a certain time in the dataset is corresponding to the user requesting the movie at that time. This assumption can be easily removed as users usually rate movies after watching them. We divide the timestamp into time periods of one hour each, assuming that MEC servers update their caching decision at an hourly basis. Fig. 2 shows the number of content requests in each time period.

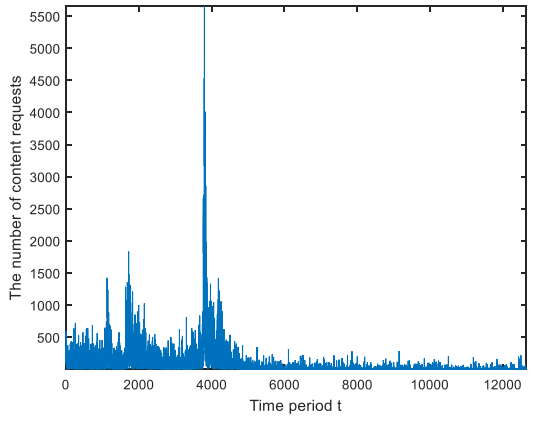

FIGURE 2. The number of content requests in each time period.

For the performance evaluation of our proposed algorithm, we classify the users into different coverage areas which served by different servers according to their Zip-code. The default number of servers M is assumed to be 5. We normalize the size of each movie file s_f to one unit and the default storage capacity of each server S_m is assumed to be 50 units. Hence, the total storage capacity in the domain corresponds to about 6% of the total number of movie files, which is a realistic assumption [12], [22]. We normalize the downloading rate between server m and data center to 1, *i.e.*, $Z_{m,0} = 1$, $m \in \mathcal{M}$ and assume the ratio of the downloading rate between server m and n to the downloading rate between server m and data center as $Z_{m,n}/Z_{m,0} = 5$, $m, n \in \mathcal{M}$, $n \neq m$. Let there be two service types 1 and 2 with weights $v_1 = 2$ and $v_2 = 1$ respectively. Therefore, service type 1 is prioritized since the associated weight of service types 1 is higher than that of service types 2. We assign the users in age 18-25 to service type 1 and classify all rest users to service type 2. Additionally, we use the average rating of movie file f as the prioritization weight w_f for movie file f.

We conduct simulation experiments to compare the performance of our multi-agent reinforcement learning (MARL) based caching scheme with the following three caching schemes. 1) Single-agent reinforcement learning (SARL) based caching scheme [32], which ignores the multi-agent nature of the problem model. Each server maintains a Q-table $Q_{m,f}$, $m \in \mathcal{M}$, $f \in \mathcal{F}$. The Q-table is updated without regard for the caching decisions made by other servers. In each time period t, the Q-table is updated by $Q_{m,f} \leftarrow Q_{m,f} + \frac{1}{C_{m,f}+1} \cdot (r_{m,f}^t - Q_{m,f}), f \in \mathcal{C}_m^t$. Then the Q-value is biased and inputted into the (α, β)-approximation oracle for computing the optimal caching decision vector. 2) Myopic caching scheme [18], which is a modified version of the least recently used (LRU) caching scheme [33]. At each replacement phase, LRU caches the most recently requested file and replaces the least recently used file. In our model, since only the content requests for those files in the local cache can be observed, LRU is inapplicable. In the Myopic caching scheme, it randomly replaces all the files that have not been requested within last time period with other files. 3) Randomized replacement (RR) caching scheme [34], which randomly places files in the local cache of MEC servers until all caches become full.

The performance metrics we use include the (cumulative) caching reward, the (cumulative) number of caching hits, the weighted average downloading latency (WADL) and the cache hit rate (CHR). The caching reward is defined as $\sum_{m \in \mathcal{M}} \sum_{f \in \mathcal{C}_m^t} r_{m,f}^t$ for time period t. The number of cache hits is defined as the number of content requests served by a server in the domain. The WADL is defined as the sum weighted downloading latency for all content requests divided by the number of content requests. The CHR is

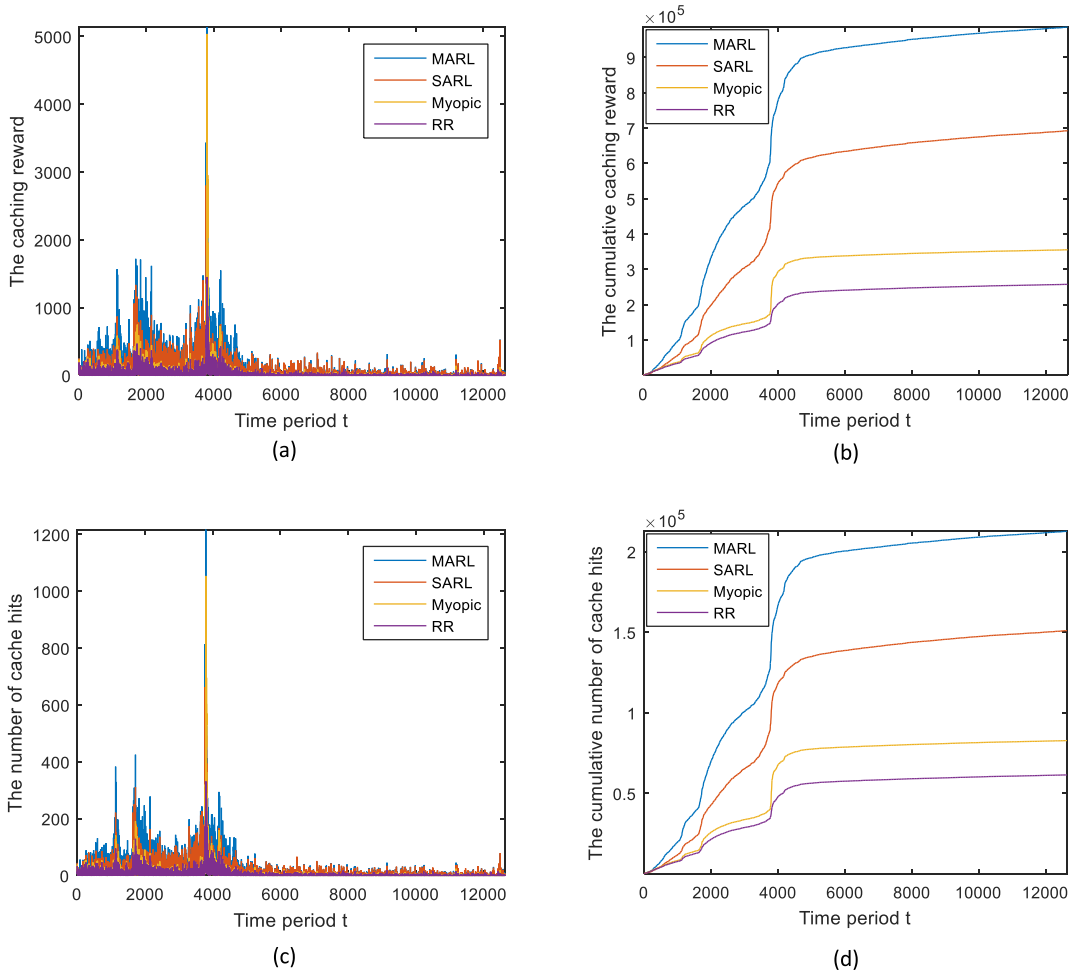

FIGURE 3. The (cumulative) caching reward and the (cumulative) number of cache hits of MARL, SARL, Myopic and RR cache schemes for different time periods. (a) The caching reward. (b) The cumulative caching reward. (c) The number of cache hits. (d) The cumulative number of cache hits.

defined as the ratio of the total number of cache hits to the number of content requests.

In the first experiment, we evaluate the (cumulative) caching reward and the (cumulative) number of cache hits over time in Fig. 3. As it is shown in Fig. 3(a) and Fig. 3(c), the caching reward and the number of caching hits of all caching schemes are bursty over time, due to the burstness of the content request process. Fig. 3(b) and Fig. 3(d) clearly illustrate that MARL and SARL caching schemes outperform Myopic and RR caching schemes in terms of the cumulative caching reward and the cumulative number of caching hits. This is because that MARL and SARL caching schemes learn from the historical users' content requests, while Myopic caching scheme learns only from one-step past and no learning is adopted in RR caching scheme. It also can be observed that the MARL caching scheme outperforms the SARL caching scheme since MARL learns the Q-values of their own caching decisions as well as those of other servers while SARL only learn the Q-values of their own caching decisions. In particular, compared with SARL, Myopic and RR caching schemes, the gain of the MARL caching scheme in terms of the cumulative number of cache hits is approximately 41%, 157% and 246%, respectively in time horizon T.

Next, we examine the WADL and CHR when the storage capacity of each server varies from 10 to 100 units in Fig. 4. As expected, the WADL decreases and the CHR increases with the increasing storage capacity of each server for all caching schemes, since more files can be cached in servers. In Fig. 4(a), it is shown that the WADL of the MARL caching scheme is significantly lower than that of SARL, Myopic and RR caching schemes. In particular, the MARL caching scheme outperforms SARL, Myopic and RR caching schemes with the improvement on the WADL approximately 8%, 21% and 24%, respectively when the storage capacity is up to 100 units.

In the following, we compare the WADL and CHR for varying number of MEC servers from 1 to 10 in Fig. 5. We can see that the WADL decreases with the number of servers in a domain for all caching schemes in Fig. 5(a), while the CHR increases with the number of servers for all caching schemes

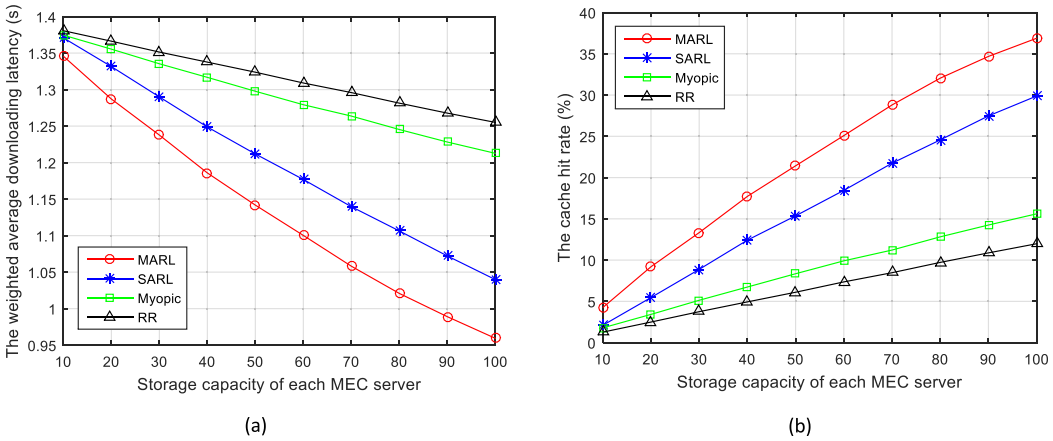

FIGURE 4. The weighted average downloading latency and the cache hit rate of MARL, SARL, Myopic and RR cache schemes for different storage capacities. (a) The weighted average downloading latency. (b) The cache hit rate.

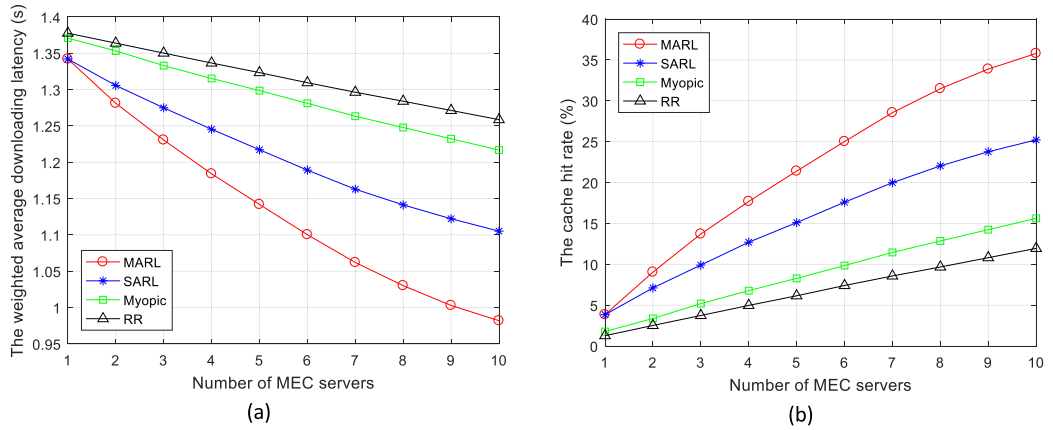

FIGURE 5. The weighted average downloading latency and the cache hit rate of MARL, SARL, Myopic and RR cache schemes for different numbers of MEC servers. (a) The weighted average downloading latency. (b) The cache hit rate.

in Fig. 5(b). This is because that with the increase of the number of servers, more users' content requests can be served in the domain. In Fig. 5(b), we can observe that the CHR of MARL caching scheme is significantly higher than that of SARL, Myopic and RR caching schemes. Compared with SARL, Myopic and RR caching schemes, the gain of MARL caching scheme in terms of CHR is approximately 41%, 129% and 199% respectively when the number of servers is up to 10. Besides, the difference becomes more significant for more MEC servers. Therefore, our proposed MARL caching scheme can greatly improve the CHR, especially for the large number of servers in a domain, by learning historical content demands.

VII. CONCLUSION

In this paper, we have addressed the cooperative content caching problem for MEC architecture, when the users' preference is unknown and only the historical content demands are observed. Considering the service differentiation, we use the weighted reduction of downloading latency as the caching reward and model the cooperative content caching problem as a multi-agent multi-armed bandit problem to maximize the total expected accumulated caching reward over a long-term horizon. We propose a MARL-based caching algorithm technically solve the problem. Q-learning is used by MEC servers to learn how to coordinate their caching decisions in multi-agent systems. MEC servers learn the Q-values of their own caching decisions in conjunction with those of other MEC servers. Since the space of Q-table is huge and thus traditional multi-agent Q-learning algorithm needs exponential number of steps to traverse all the Q-values, a combinatorial upper confidence bound method is proposed to reduce the space of the Q-table to effectively reduce the complexity. Moreover, we adopt the belief-maintenance procedures to estimate other servers' strategies. Simulation results show that the proposed MARL-based caching scheme can significantly reduce content downloading latency and improve content cache hit rate in comparison with other popular caching schemes.

REFERENCES

[1] *Cisco Visual Networking Index: Forecast and Methodology, 2016–2021*. Accessed: Nov. 21, 2017. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html
[2] L. Qiu and G. Cao, "Popularity-aware caching increases the capacity of wireless networks," in *Proc. IEEE INFOCOM*, May 2017, pp. 1–9.
[3] Y. Zhou, L. Chen, C. Yang, and D. M. Chiu, "Video popularity dynamics and its implication for replication," *IEEE Trans. Multimedia*, vol. 17, no. 8, pp. 1273–1285, Aug. 2015.
[4] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung, "Cache in the air: Exploiting content caching and delivery techniques for 5G systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 131–139, Feb. 2014.
[5] A. Passarella, "A survey on content-centric technologies for the current Internet: CDN and P2P solutions," *Comput. Commun.*, vol. 35, no. 1, pp. 1–32, 2012.
[6] 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.
[7] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.
[8] M. T. Beck, M. Werner, S. Feld, and S. Schimper, "Mobile edge computing: A taxonomy," in *Proc. 6th Int. Conf. Adv. Future Internet*, Jan. 2014, pp. 48–55.
[9] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.
[10] Y. Li, Y. Xu, T. Lin, X. Wang, and S. Ci, "A novel coordinated edge caching with request filtration in radio access network," *Sci. World J.*, vol. 2013, Art. no. 654536.
[11] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.
[12] K. Poularakis, G. Iosifidis, and L. Tassiulas, "Approximation algorithms for mobile data caching in small cell networks," *IEEE Trans. Commun.*, vol. 62, no. 10, pp. 3665–3677, Oct. 2014.
[13] M. Gregori, J. Gómez-Vilardebó, J. Matamoros, and D. Gündüz, "Wireless content caching for small cell and D2D networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1222–1234, May 2016.
[14] R. Amer, M. M. Butt, M. Bennis, and N. Marchetti, "Inter-cluster cooperation for wireless D2D caching networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 9, pp. 6108–6121, Sep. 2018.
[15] W. Jiang, G. Feng, and S. Qin, "Optimal cooperative content caching and delivery policy for heterogeneous cellular networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 5, pp. 1382–1393, May 2017.
[16] E. Baştuğ, M. Bennis, and M. Debbah, "A transfer learning approach for cache-enabled wireless networks," in *Proc. Int. Symp. Model. Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt)*, May 2015, pp. 161–166.
[17] T. Hou, G. Feng, S. Qin, and W. Jiang, "Proactive content caching by exploiting transfer learning for mobile edge computing," *Int. J. Communication Syst.*, vol. 31, no. 11, Jul. 2018, Art. no. e3706.
[18] P. Blasco and D. Gündüz, "Learning-based optimization of cache content in a small cell base station," in *Proc. IEEE Int. Conf. Commun.(ICC)*, Jun. 2014, pp. 1897–1903.
[19] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, "Learning distributed caching strategies in small cell networks," in *Proc. 11th Int. Symp. Wireless Commun. Syst. (ISWCS)*, Aug. 2014, pp. 917–921.
[20] J. Song, M. Sheng, T. Q. S. Quek, C. Xu, and X. Wang, "Learning-based content caching and sharing for wireless networks," *IEEE Trans. Commun.*, vol. 65, no. 10, pp. 4309–4324, Oct. 2017.
[21] D. Liu and C. Yang. (Oct. 2017). "Caching at base stations with heterogeneous user demands and spatial locality." [Online]. Available: https://arxiv.org/abs/1710.09983
[22] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 1024–1036, Feb. 2017.
[23] Y. Li, J. Liu, B. Cao, and C. Wang, "Joint optimization of radio and virtual machine resources with uncertain user demands in mobile cloud computing," *IEEE Trans. Multimedia*, vol. 20, no. 9, pp. 2427–2438, Sep. 2018.
[24] J. Shen, S. Suo, H. Quan, X. Zhao, H. Hu, and Y. Jiang, *3GPP Long Term Evolution: Principle and System Design*. Beijing, China: Posts Telecom Press, 2008.
[25] W. Jiang, G. Feng, S. Qin, and T. S. P. Yum, "Efficient D2D content caching using multi-agent reinforcement learning," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2018, pp. 511–516.
[26] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2, pp. 235–256, 2002.
[27] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework and applications," in *Proc. Int. Conf. Mach. Learn.*, Feb. 2013, pp. 151–159.
[28] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," in *Proc. AAAI/IAAI*, Jul. 1998, pp. 746–752.
[29] A. A. Korbut and I. K. Sigal, "Exact and greedy solutions of the knapsack problem: The ratio of values of objective functions," *J. Comput. Syst. Sci. Int.*, vol. 49, no. 5, pp. 757–764, 2010.
[30] 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.
[31] *MovieLens 1M Dataset*. Accessed: Dec. 9, 2015. [Online]. Available: http://grouplens.org/datasets/movielens/1m/
[32] W. Jiang, G. Feng, S. Qin, T. S. P. Yum, and G. Cao, "Multi-agent reinforcement learning for efficient content caching in mobile D2D networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 3, pp. 1610–1622, Mar. 2019.
[33] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. 18th Annu. Joint Conf. IEEE Comput. Commun. Soc. (INFOCOM)*, vol. 1, Mar. 1999, pp. 126–134.
[34] K. Psounis and B. Prabhakar, "A randomized Web-cache replacement scheme," in *Proc. IEEE INFOCOM*, vol. 3, Apr. 2001, pp. 1407–1415.

WEI JIANG received the B.S. degree from the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications (CQUPT), in 2013. She is currently pursuing the Ph.D. degree with the School of Communication and Information Engineering, University of Electronic Science and Technology of China (UESTC). She was a Visiting Ph.D. Student with the Pennsylvania State University (PSU), from 2017 to 2018. Her current research interests include next-generation mobile communication systems, machine learning and content caching, and distribution techniques in mobile networks.

GANG FENG (M'01–SM'06) received the B.Eng. and M.Eng. 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 2000, as an Assistant Professor and was promoted as an Associate Professor, in 2005. He is currently a Professor with the National Laboratory of Communications, UESTC. He 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, and next-generation cellular networks.

SHUANG QIN received the B.S. degree in electronic information science and technology and the Ph.D. degree in Communication and Information System from the University of Electronic Science and Technology of China (UESTC), in 2006 and 2012, respectively. He is currently an Associate Professor with the National Key Laboratory of Science and Technology on Communications, UESTC. His research interests include cooperative communication in wireless networks and data transmission in opportunistic networks.

YIJING LIU received the B.S. degree from the School of Communication and Information Engineering, Chongqing University of Post and Telecommunications (CQUPT), in 2017. She is currently pursuing the M.S. degree with the National Key Laboratory of Science and Technology on Communication, University of Electronic Science and Technology of China (UESTC). Her current research interests include next generation mobile networks, mobile edge computing, and network function parallelization.