Deep Reinforcement Learning for Cooperative Edge Caching in Future Mobile Networks

Ding Li[1], Yiwen Han[1], Chenyang Wang[1], GaoTao Shi[1], Xiaofei Wang[1], Xiuhua Li[2], and Victor C. M. Leung[3]
[1]College of Intelligence and Computing,Tianjin University, Tianjin, China
[2]School of Big Data & Software Engineering, Chongqing University, Chongqing, China
[3]Dept. Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada
{liding_cs, hanyiwen, chenyangwang, shgt, xiaofeiwang}@tju.edu.cn, lixiuhua1988@gmail.com,vleung@ece.ubc.ca

***Abstract*—To satisfy rapidly increasing multimedia service requests from mobile users, content caching at the network edges (e.g., base stations) has been regarded as a promising technique in future mobile networks. In this paper, by virtue of Deep Reinforcement Learning (DRL) with respect to solving complicated control problems, we propose a framework on Double Deep Q-Network for cooperative edge caching in mobile networks. Particularly, we aim at minimizing the long-term average content fetching delay of mobile users without requiring any priori knowledge of content popularity distribution. Trace-driven simulation results show that our proposed framework outperforms some existing caching algorithms, including Least Recently Used (LRU), Least Frequently Used (LFU) and First-In First-Out (FIFO) caching strategies by 7%, 11% and 9% improvements, respectively. Besides, our proposed work is further shown that only average 4% performance loss exists compared to an omniscient oracle algorithm.**

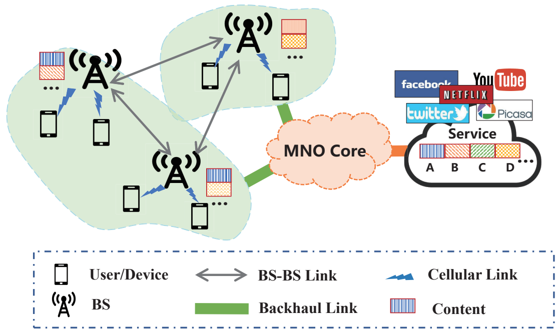


Fig. 1. Architecture of edge caching within cooperative base stations.

I. Introduction

With the rapid development of wireless access technology and mobile devices, Internet services and applications are gradually migrating to mobile networks. As a result, current mobile networks are required to support a growing number of multimedia service requests from mobile users carrying mobile devices (e.g., smart phones and tablets), especially for accessing high-quality contents (e.g., audio, photos and video). This explosive growth of wireless content services results in the rapid increase of network traffic load and degradation in Quality of Service (QoS) for users, which poses an enormous challenge to future mobile networks.

However, as indicated in [1], many popular contents are requested frequently by different users, and thus how to effectively reduce duplicated backhual traffic by offloading the traffic via local short-range communications has become a hot research topic. Content caching is a promising technique that could put computation and storage services away from remote server to the edges, e.g., Base Stations (BSs), of mobile networks. Consequently, caching contents in BSs can bring the requested contents much closer to mobile users in the routing distance of the network topology, instead of excessively downloading duplicated contents from service providers (SPs) via backhaul networks [2], [3].

There have been numerous studies focusing on content caching at the edges of mobile networks. For instance, studies in [4]–[6] discussed the potentials of employing caching technique in mobile networks. Besides, FemtoCaching [7], [8] and AMVS-NDN [9], focused on caching popular contents in BSs, evolved NodeBs (eNBs) or femtocells. The studies in [8], [10], [11] proposed strategies of collaborative caching in BSs to improve the QoS of users especially on access delay. However, these studies have been conducted on finding an optimal/suboptimal solution via traditional optimization techniques, which are often lack of the self-adaption in dynamic environments and need nearly global information that is hard to achieve in the real-world systems.

Recently, artificial intelligence (AI) is getting more attention in wireless communications. The studies in [12], [13] showed that reinforcement learning (RL) has great potentials used for the scheme design of content caching in BSs. Specifically, the authors of [12] proposed a cache replacement strategy based on Q-Learning to reduce traffic load in future cellular networks, and RL was also employed for cache placement [13] by using multi-armed bandit (MAB). However, traditional RL techniques are not feasible for the practical network environments where the state-action space is tremendous. To make up this disadvantage, the technique of deep reinforcement learning (DRL) was used for content caching such as in [14]. However, employing DRL for the scheme design of content caching in wireless networks is not explored well, especially for improving users' QoS and offloading network traffic and satisfying content requests.

In this paper, we focus on the rational decision making for cache replacement and requests routing among BSs, aiming to reduce the access delay of users and the traffic burden on the backhual. For coping with the dynamic wireless environments and content requests, also with the aim of caching optimization in long-term fashion, we propose a framework based on DRL for cooperative content caching and request routing among BSs. The main contributions of this paper are summarized below:

- We investigate the issues of cooperative edge caching and request routing in mobile networks where contents are cached in BSs, thereby offloading duplicated traffic and improving both the specific QoS (e.g., access delay).
- We model the content caching and request routing problem as a Markov Decision Process (MDP) and deploy a learning framework based on Double Deep Q-Network (DQN) for cooperative content caching among BSs.
- Experimental results based on large-scale realistic traces show that the proposed framework achieves better performance compared to the existing schemes including LRU, LFU and FIFO, the performance enhancements are 7%, 11% and 9%, respectively. We analytically bound the loss of the our proposed work compared to an oracle, which is shown that only average 4% performance loss exists.

The remainder of this article is organized as follow. We first describe the system model and formulate the problem in Sec. II. The details of caching policy design are presented in Sec. III. Further, trace-driven simulation results are shown in Sec. IV. Finally, Sec. V concludes the paper.

II. System Model and Problem Formulation

In this section, we first introduce the general cooperative edge caching architecture and topology. Then, we discuss the communication model. Lastly, we formulate the problem of caching replacement within the MDP framework.

A. Cooperative Edge Caching Architecture and Topology

The general system architecture of cooperative edge caching in mobile networks is illustrated in Fig. 1. We consider a small cellular coverage area consisting of fully connected N_b BSs (denoted as a set $\mathcal{N} = \{1, 2, ..., N_b\}$) and N_u users (denoted as a set $\mathcal{U} = \{1, 2, ..., N_u\}$). Each BS has a finite cache size C. We denote $\mathcal{F} = \{1, 2, ..., F\}$ as popular contents requested during a relatively long time, and $\{s_j\}_{F\times 1}$ as their sizes. Denote $(P_f)_{F\times 1}$ is the probability of the request of content f to the requests of all the content in the network. The content popularity is modeled by a MZipf distribution as [18], [19]

$$P_f = \frac{(R_f + \tau)^{-\beta}}{\sum_{i\in\mathcal{F}} (R_i + \tau)^{-\beta}}, \ \forall f \in \mathcal{F}, \tag{1}$$

where the R_f is popularity index of content f in a descending order, τ is the plateau factor and β is the skewness factor.

Each user is served by the local BS via cellular links. Particularly, if the requested content is not cached in the local BS, the local BS will fetch it from its cooperative/neighboring BSs or directly download it from SPs via the mobile network operator core through backhaul links.

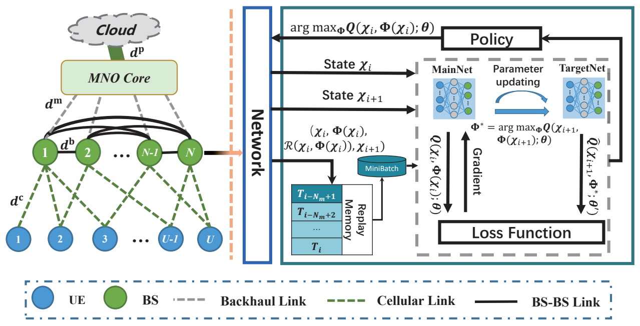


Fig. 2. Training Process in a BS.

B. Communication Model

The content fetching delay of users (we simply use delay instead hereafter) can be defined as the round-trip time for obtaining a requested content by a user equipment (UE) via the MNO core, which is formed as shown in Fig. 2. It consists of the transmission delay d^c from the BS to the UE, the delay d^b between BSs, the delay d^m between the BS and the MNO network,and the delay d^p between the BS and the SP. We model the wireless transmission delay between the UE and the BS as the ratio between the content size to the downlink data rate. Similar to [15], there are M wireless channels and the set of channels is denoted as $\mathcal{M} = \{1, 2, ..., M\}$. We denote $a_u \in \mathcal{M}$ as the channel allocation decision of the BS to mobile user u. The BS chooses a wireless channel a_u to communicate to User u . The downlink data rate from BS n to UE u can be expressed as

$$r_{u,n} = w \log_2 \left(1 + \frac{q_u g_{u,n}}{\sigma^2 + \sum_{v\in\mathcal{U}\setminus\{u\}:a_v=a_u} q_v g_{v,n}} \right), \tag{2}$$

where w is channel bandwidth, σ^2 represents the background noise power, q_u is transmission power of BS n to UE u, $g_{u,n}$ is the channel gain and determined by the distance (denoted as $l_{u,n}$) between the UE u and the BS n. We assume that the communications of BS-BS, BS-MNO and BS-SP are via fibers, and we can derive the wireless transmission delay as $d^\mathrm{c} = \{s_j\}_{F\times 1} / \ r_{u,n}$.

C. Cache Replacement Model

We model the cache replacement process as a MDP. Besides, we discuss the details of the related state space, action space and reward function as follow.

1) ***State Space:*** We define $s^\mathrm{c}_{i,f}$ as the content caching state during each decision epoch i with respect to the content $f \in \mathcal{F}$, which independently picks a value from a state space $\mathbb{P}$. $s^\mathrm{c}_{i,f} = 1$ means content f is cached in the BS and $s^\mathrm{c}_{i,f} = 0$ means the opposite. In addition, s^r_i is introduced to denote the current requesting content from the UE in the decision epoch i. The state of an available BS during each decision epoch i can be represented by

$$\chi_i = (s^\mathrm{r}_i, \mathbf{s}^\mathrm{c}_i) \in \mathcal{X} \overset{\text{def}}{=} \{1, 2, .., F\} \times \{\times_{f\in\mathcal{F}} \mathbb{P}\}, \tag{3}$$

where $\mathbf{s}^\mathrm{c}_i = (s^\mathrm{c}_{i,f} : f \in \mathcal{F})$.

*2) **Action Space:*** The system action with respect to the state $\boldsymbol{\chi}_i$ can be denoted as $\boldsymbol{\Phi}(\boldsymbol{\chi}_i)$. All cooperative BSs possess the same action space $\mathcal{A}$ as

$$\mathcal{A} = \{\mathbf{a}_i^{\text{local}}, \mathbf{a}_i^{\text{co}-\text{BS}}, a_i^{\text{SP}}\}. \tag{4}$$

Namely, the system action $\boldsymbol{\Phi}(\boldsymbol{\chi}_i)$ can be divided into three parts according to their different characters as following:

*a) **Requests Handled by the Local BS:*** The available cache control in the local BS is represented by $\mathbf{a}_i^{\text{local}} \overset{\text{def}}{=} [a_{i,0}^{\text{local}}, a_{i,1}^{\text{local}}, \ldots, a_{i,F}^{\text{local}}]$, where $a_{i,f}^{\text{local}} \in \{0,1\} (f \in 1, ..., F)$ indicates that whether and which content in the local BS should be replaced by the current requesting content, ($a_{i,0}^{\text{local}} \in 0, 1$) represents that whether the local BS make replacement, i.e, the content request is handled by the BS itself.

*b) **Requests Handled by Cooperative BSs:*** Though the local BS could take in charge of the requesting content, it can also perform action $\mathbf{a}_i^{\text{co}-\text{BS}} \overset{\text{def}}{=} [a_{i,1}^{\text{co}-\text{BS}}, \ldots, a_{i,N}^{\text{co}-\text{BS}}]$ to instruct which cooperated BS is selected to take over its own task. Herein, each element $a_{i,n}^{\text{co}-\text{BS}} \in \{0,1\}$, and specifically $a_{i,n}^{\text{co}-\text{BS}} = 1$ means that the BS n is chosen to handle the current request.

*c) **Requests Handled by SPs:*** Certainly, each BS can further directly relay requests of UEs to SPs and $a_i^{\text{SP}} \in \{0,1\}$ is introduced to represent this kind of action, where $a_i^{\text{SP}} = 1$ means that the request is chosen to be directly handled by SPs, viz., the UE shall fetch the content from SPs.

*3) **Reward Function:*** Reward (utility) function $\mathcal{R}(\boldsymbol{\chi}, \boldsymbol{\Phi})$, which determines the reward fed back to the BS when performing the action $\boldsymbol{\Phi}(\boldsymbol{\chi}_i)$ upon the state $\boldsymbol{\chi}_i$, shall be determined in the interactive wireless environment to lead the DRL agent (we will introduce it later) in BSs toward achieving ideal performance. From the perspective of MNOs, it is important to satisfy the QoS requirement of mobile users while reducing the traffic load via cooperative BS caching (namely edge caching). Among QoS metrics, the most important one is delay for users to access the demanded contents. Our objective is to minimize the average delay experienced by users. Thus, in our edge caching architecture, we devise the reward function as

$$\mathcal{R}(\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i)) = \begin{cases} \mathrm{e}^{-(d^{\text{c}})}, & \boldsymbol{\Phi}(\boldsymbol{\chi}_i) = \mathbf{a}_i^{\text{local}} \\ \mathrm{e}^{-(d^{\text{c}}+d^{\text{b}})}, & \boldsymbol{\Phi}(\boldsymbol{\chi}_i) = \mathbf{a}_i^{\text{co}-\text{BS}}, \\ \mathrm{e}^{-(d^{\text{c}}+d^{\text{m}}+d^{\text{p}})}, & \boldsymbol{\Phi}(\boldsymbol{\chi}_i) = a_i^{\text{SP}}, \end{cases} \tag{5}$$

where negative exponential function with respect to the delay is adopted here to guide the objective of minimizing the average delay.

D. Problem Formulation

Based on (5), The expected long-term reward value conditioned on any initial state $\boldsymbol{\chi}_1$ can be expressed as

$$V(\boldsymbol{\chi}, \boldsymbol{\Phi}) = \mathbb{E}_{\boldsymbol{\Phi}} \left[\sum_{i=1}^{\infty} (\gamma)^{i-1} \cdot \mathcal{R}(\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i)) | \boldsymbol{\chi}_1 = \boldsymbol{\chi} \right], \tag{6}$$

where $\gamma \in [0, 1)$ is the discount factor, and $V(\boldsymbol{\chi}, \boldsymbol{\Phi})$ is named as the state value function for the BS in the state $\boldsymbol{\chi}$ under policy $\boldsymbol{\Phi}$.

Each BS is expected to learn an optimal control policy $\boldsymbol{\Phi}^*$ that maximizes $V(\boldsymbol{\chi}, \boldsymbol{\Phi})$ with any initial state $\boldsymbol{\chi}$. The optimization objective is defined as

$$R^{\text{long}} = \max \mathbb{E}_{\boldsymbol{\Phi}} \left[\sum_{i=1}^{\infty} (\gamma)^{i-1} \cdot \mathcal{R}(\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i)) | \boldsymbol{\chi}_1 = \boldsymbol{\chi} \right], \tag{7}$$

The optimal control policy can be described as follows

$$\boldsymbol{\Phi}^* = \underset{\boldsymbol{\Phi}}{\text{argmax}}\, V(\boldsymbol{\chi}, \boldsymbol{\Phi}), \forall \boldsymbol{\chi} \in \mathcal{X}. \tag{8}$$

Further, we can obtain the optimal state value function $V(\boldsymbol{\chi})$ for any initial state $\boldsymbol{\chi}$ as

$$V(\boldsymbol{\chi}) = V(\boldsymbol{\chi}, \boldsymbol{\Phi}^*), \forall \boldsymbol{\chi} \in \mathcal{X}. \tag{9}$$

III. Learning based Cache Replacement

A. Reinforcement Learning

RL is a type of machine learning and it focuses on enabling the agent to automatically determine the ideal behaviour within a specific context to maximize its cumulative expected return. RL problems can be described as the optimal control decision making problem in MDP. Q-Learning, which is one of effective model-free RL algorithms, is first adopted here to be analyzed for the introduction of our proposed method. In our cooperative edge caching architecture, the agent pertained to the BS senses and obtains its current cache state $\boldsymbol{\chi}_i$. Then, the agent selects and carries out an action $\boldsymbol{\Phi}(\boldsymbol{\chi}_i)$. Meantime, the environment experiences a transition from $\boldsymbol{\chi}_i$ to a new state $\boldsymbol{\chi}_{i+1}$ and obtains a reward $\mathcal{R}(\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i))$.

According to the Bellman Equation, the optimal state-value function $V(\boldsymbol{\chi})$ can be expressed as (10), where $\boldsymbol{\chi} = \boldsymbol{\chi}_i$ is the state at current decision epoch i, and the next state is $\boldsymbol{\chi}' = \boldsymbol{\chi}_{i+1}$ after taking the action $\boldsymbol{\Phi} = \boldsymbol{\Phi}(\boldsymbol{\chi}_i)$.

$$V(\boldsymbol{\chi}) = \max_{\boldsymbol{\Phi}} \left\{ \mathcal{R}(\boldsymbol{\chi}, \boldsymbol{\Phi}) + \gamma \cdot \sum_{\boldsymbol{\chi}'} \Pr\{\boldsymbol{\chi}' | \boldsymbol{\chi}, \boldsymbol{\Phi}\} \cdot V(\boldsymbol{\chi}') \right\}. \tag{10}$$

Commonly, right-hand side of (10) can be further abstracted as

$$Q(\boldsymbol{\chi}, \boldsymbol{\Phi}) = \mathcal{R}(\boldsymbol{\chi}, \boldsymbol{\Phi}) + \gamma \cdot \sum_{\boldsymbol{\chi}'} \Pr\{\boldsymbol{\chi}' | \boldsymbol{\chi}, \boldsymbol{\Phi}\} \cdot V(\boldsymbol{\chi}'). \tag{11}$$

It is simple to obtain that the optimal state value function $V(\boldsymbol{\chi})$ could be expressed as

$$V(\boldsymbol{\chi}) = \max_{\boldsymbol{\Phi}} Q(\boldsymbol{\chi}, \boldsymbol{\Phi}). \tag{12}$$

By incorporating (12), (11) can be rewritten as

$$\begin{aligned} Q(\boldsymbol{\chi}, \boldsymbol{\Phi}) = &\mathcal{R}(\boldsymbol{\chi}, \boldsymbol{\Phi}) + \\ &\gamma \cdot \sum_{\boldsymbol{\chi}'} \Pr\{\boldsymbol{\chi}' | \boldsymbol{\chi}, \boldsymbol{\Phi}\} \cdot \max_{\boldsymbol{\Phi}'} Q(\boldsymbol{\chi}', \boldsymbol{\Phi}'). \end{aligned} \tag{13}$$

At last, the iterative formula of Q-function can be obtained as

$$\begin{aligned} &Q^{i+1}(\boldsymbol{\chi}, \boldsymbol{\Phi}) = Q^i(\boldsymbol{\chi}, \boldsymbol{\Phi}) + \\ &\alpha^i \cdot \left(\mathcal{R}(\boldsymbol{\chi}, \boldsymbol{\Phi}) + \gamma \cdot \max_{\boldsymbol{\Phi}'} Q^i(\boldsymbol{\chi}', \boldsymbol{\Phi}') - Q^i(\boldsymbol{\chi}, \boldsymbol{\Phi}) \right), \end{aligned} \tag{14}$$

where $\alpha^i \in [0, 1)$ is the learning rate and the state $\boldsymbol{\chi}_i$ will turn to the state $\boldsymbol{\chi}_{i+1}$ when the agent chooses action $\boldsymbol{\Phi}(\boldsymbol{\chi}_i)$ along with the corresponding reward $R(\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i))$. Based on (14), the Q-Table can be used to store the Q value of each state-action pair when the state and action space dimensions are not high in the Q-Learning algorithm. However, the learning process in Q-Learning becomes extremely slow when the scenarios are with huge network states and action spaces. Thus, deep learning techniques are taken hereafter as a potential feasible method to approximate the value function.

B. Double Deep Q-Learning

Deep Q-Learning is effective in solving complicated and high dimensional RL problems, and the Multi-Layer Perceptron (MLP) structure adopted in it can be used to approximate the optimal state-action Q-function.

In our model, we use a robust version of DQN, namely Double DQN [17], to train our DRL agents in BSs. The Q-function could be approximated to the optimal Q value by updating the parameter $\boldsymbol{\theta}_i$ of MLP as follows

$$Q(\boldsymbol{\chi}, \boldsymbol{\Phi}) \approx Q((\boldsymbol{\chi}, \boldsymbol{\Phi}); \boldsymbol{\theta}_i). \tag{15}$$

In addition, each DRL agent owns a transition memory, also named experience replay, with a finite size N_m to store experienced transitions. The transition sample can be denoted as $T_i = (\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i), \mathcal{R}(\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i)), \boldsymbol{\chi}_{i+1})$, which represents one state transition. The whole experience pool can be represented as $\mathcal{M} = \{T_{i-N_\mathrm{m}+1}, ..., T_i\}$ and each agent updates $\mathcal{M}$ with the most recent experienced transitions. Besides, about the characteristic of Double DQN, each DRL agents maintains two Q networks, viz. $Q(\boldsymbol{\chi}, \boldsymbol{\Phi}; \boldsymbol{\theta}_i)$ and $\hat{Q}(\boldsymbol{\chi}, \boldsymbol{\Phi}; \boldsymbol{\theta}'_i)$, with network Q is used to choose action and network $\hat{Q}$ to evaluate action. Noted that, the weight parameters $\boldsymbol{\theta}'_i$ of network $\hat{Q}$ are updated periodically by the counterpart $\boldsymbol{\theta}_i$ of network Q. Throughout the training process, the DRL agent randomly samples a mini-batch $\tilde{\mathcal{M}}$ from the experience replay $\mathcal{M}$. Then at each epoch, the network Q is trained towards the direction of minimizing the loss function as

$$L(\boldsymbol{\theta}_i) = \mathbb{E}_{(\boldsymbol{\chi}, \boldsymbol{\Phi}, \mathcal{R}(\boldsymbol{\chi}, \boldsymbol{\Phi}), \boldsymbol{\chi}') \in \tilde{\mathcal{M}}_i} \Big[\Big(\mathcal{R}(\boldsymbol{\chi}, \boldsymbol{\Phi}) + \gamma \cdot \hat{Q}\Big(\boldsymbol{\chi}', \arg\max_{\boldsymbol{\Phi}'} Q(\boldsymbol{\chi}', \boldsymbol{\Phi}'; \boldsymbol{\theta}_i); \boldsymbol{\theta}'_i\Big) - Q(\boldsymbol{\chi}, \boldsymbol{\Phi}; \boldsymbol{\theta}_i)\Big)^2 \Big]. \tag{16}$$

And with (16), the gradient guiding updates of $\boldsymbol{\theta}$ can be calculated by $\frac{\partial L(\boldsymbol{\theta}_i)}{\partial \boldsymbol{\theta}_i}$. Hence, Stochastic Gradient Descent (SGD) is performed until the convergence of Q networks for approximating optimal state-action Q-function. The training algorithm based on the Double DQN in Algorithm 1.

Algorithm 1 Double DQN-based Content Caching Algorithm

```
Initialization:
    Initialize experience replay memory $\mathcal{M}$.
    Initialize main $Q$ network with random weights $\boldsymbol{\theta}$.
    Initialize target $\hat{Q}$ network with $\boldsymbol{\theta}' = \boldsymbol{\theta}$.
Iteration:
 1: for episode $i = 1$ to $I$ do
 2:     Generate $p$ at random
 3:     if $p \le \varepsilon$
 4:         randomly select an action
 5:     else
 6:         select action $\arg\max_{\boldsymbol{\Phi}(\boldsymbol{\chi}_i)} Q(\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i); \boldsymbol{\theta}_i)$
 7:     Execute action $\boldsymbol{\Phi}(\boldsymbol{\chi}_i)$, obtain immediate reward
        $\mathcal{R}(\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i))$ and observe the new state $\boldsymbol{\chi}_{i+1}$.
 8:     Construct $T_i = (\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i), \mathcal{R}(\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i)), \boldsymbol{\chi}_{i+1})$.
 9:     Store the transition $T_i$ into $\mathcal{M}$.
10:     Sample a random mini-batch of transitions $\tilde{M}_i \in \mathcal{M}$.
11:     Update the main $Q$ network parameter $\boldsymbol{\theta}_i$ with $\frac{\partial L(\boldsymbol{\theta}_i)}{\partial \boldsymbol{\theta}_i}$.
12:     Update the target $\hat{Q}$ network parameter $\boldsymbol{\theta}'_i$ periodically.
13: end for
```

IV. Trace-driven Simulation Results

A. Simulation Settings

In our simulations, four BSs are employed with maximum cover range 250 m, $g_{u,n} = 30.6 + 36.7 \log_{10} l_{u,n}$ in dB [16] is taken as the channel gain mode. Each BS consists of $M = 20$ channels and the channel bandwidth is set as 20 MHz. Besides, the total transmit power of BS is 40 W with serving at most 300 UEs. With respect to the parameter settings of Double DQN, a single-layer fully-connected feed forward neural network, which includes 200 neurons, is used to serve as the target and the eval Q network. Other parameter values are given in Table I.

TABLE I
PARAMETER VALUES

Symbol	Range	Description
F	10,000	Number of files
w	20 MHz	Channel bandwidth
σ^2	-95 dBm	Noise power
d^b	20 ms	The delay between BSs
d^p	200 ms	The delay between the BS and the SP
$\mathcal{M}$	5000	Replay memory capacity
s_f	(0,8] Mbit	Size of each file
$\tilde{\mathcal{M}}_i$	200	Minibatch size
γ	0.9	Discount factor
ϵ	0.1	Exploration probability
ς	0.05	Learning rate
ϕ	250	The period of replacing target Q network

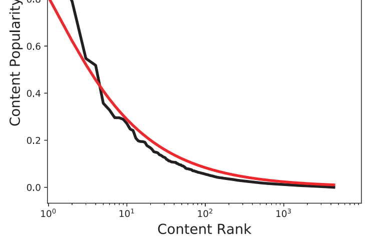
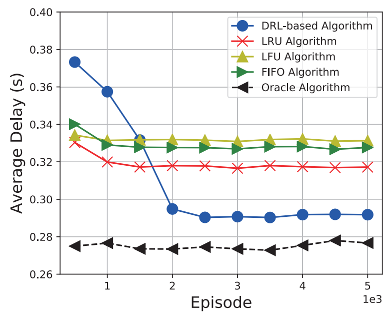
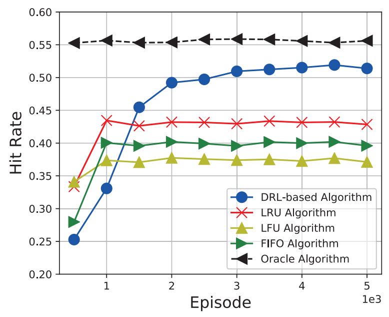


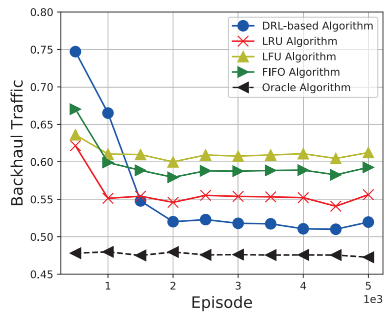
Fig. 3. Content Popularity



(a) Performance comparison of average delay.

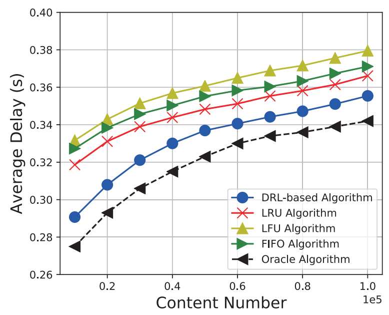


(b) Performance comparison of hit rate.

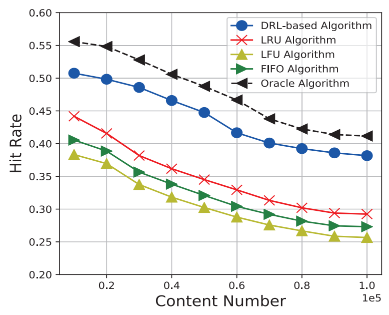


(c) Performance comparison of traffic.

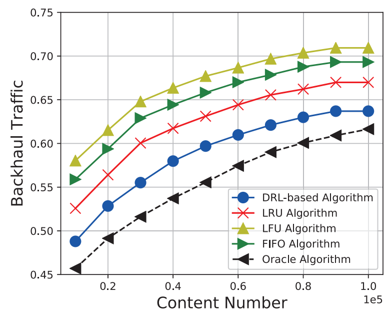
Fig. 4. Performance evaluation in terms of average delay, hit rate and traffic with respect to the time



(a) Average delay with different content number.



(b) Hit rate with different content number.



(c) Backhaul traffic with different content number

Fig. 5. Performance evaluation in terms of average delay, hit rate and traffic with respect to content numbers

$Xender$ is a popular mobile application for D2D sharing and largely used in India. We capture the sharing data from $Xender$ for one month (from $01/08/2016$ to $31/08/2016$), including $450,786$ active mobile users, $153,482$ content files, and $271,785,952$ content requests [18]. As shown in Fig. 3, the content popularity distribution in the $Xender's$ trace can be fitted by a MZipf distribution with a plateau factor of -0.34 and a skewness factor of 0.55. Particularly, 1200 mobile users which frequently request small files from the $Xender's$ trace are extracted. Thus, in the follows, large-scale simulations based on the $Xender's$ trace are conducted 50 times for averaging.

B. Evaluation Results

To evaluate the performance of our algorithm, comparisons with other caching strategies are provided.

1) LRU: Replace the least recently used content first.

2) LFU: Replace the least-frequently used content first.

3) FIFO: Replace the first-in content first.

4) Oracle [20]: An impractical omniscient oracle algorithm possessing the optimal performance, which assumes a priori knowledge about the complete behavior of UEs.

Fig. 4 shows the performance comparison in terms of average delay, hit rate and traffic with total $F = 10,000$ contents and $C = 100$ MB cache size. Certainly, Oracle outperforms other strategies on all metrics, which is approximately the optimal solution, since it assumes the whole information of user preferences is obtained in advance. However, Oracle is not practical actually since the future information can not be acquired. We can observe that the average delay of the proposed algorithm is quite high and does not show its advantage at the beginning. But soon the average delay decreases and then maintains a relatively stable value, approaching the value of Oracle, with the execution of training processes.

The effectiveness of the proposed algorithm can be attributed to that the reward function is designed for reducing the user delay, which make the DRL agent march toward minimizing the average delay. During the learning process, our DRL agent can learn the multi-dimensional features in popularity patterns with the increase of training episodes. In spite of Oracle, the proposed algorithm is better than that of LRU, LFU and FIFO, specifically, it can decrease the average delay up to 35% compared to the situation where no caching policies are adopted, with LRU, FIFO and LFU achieve 29%, 27% and 26% respectively.

Similarly, the delay optimization has positive impacts on metrics of the hit rate and the traffic offload. The proposed work improves the cache hit rate by up to 8%, 10% and 15% when compared with LRU, LFU and FIFO.

Fig. 4(c) compares the performance of different caching strategies in terms of relieving the backhaul, which also means that more traffics are offloaded in the BSs rather than SPs via backhaul. It is obvious that LRU, LFU and FIFO strategies are

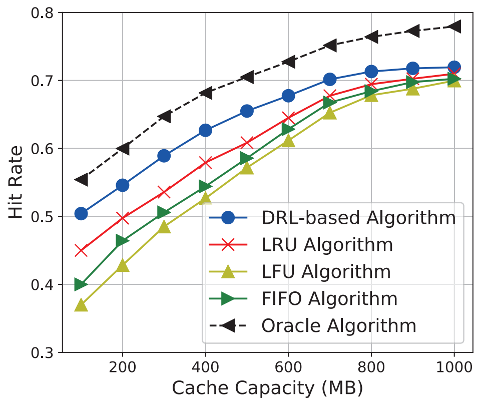


Fig. 6. Performance of hit rate under different cache capacity

inferior to our work, while LFU has the worst performance.

Fig. 5 shows the performance comparison under different content numbers. In Fig. 5(a)-5(c), with varying the content number from $10,000$ to $100,000$ by fixing the cache capacity $C = 100$ MB, it can be inferred that our work scales well with the content number and can still achieve better performance even when massive of contents are considered in the scenario. Besides, when the cache capacity of BSs is fixed, more popular contents means more contents should be cached with the increase of content number, which giving rise to frequent cache replacement. Meanwhile, the hit rate decreases with the increase of the total content numbers as shown in Fig. 5(b). From this point of view, an appropriate cache capacity may be also a key point for the architecture of edge caching.

Hence, in Fig. 6, different algorithms are compared with respect to cache capacity. It can be seen that the hit rate increases with the cache capacity growing from 100 MB to 700 MB. However, the hit rate maintains a relatively stable value from 700 MB to 1000 MB (with higher cache capacity), the reason of which lies in that when the cache capacity is big enough to store more contents, replacement processes rarely happen. In addition, variation trends of average delay and the traffic on backhaul also demonstrate our analysis below. It is worth noting that a specific value of cache capacity is only taken for demonstrate the effectiveness of our work. Practically, each BS can be equipped with larger cache space to face massive popular contents in the real wireless environment.

V. CONCLUSION

In this paper, we propose a DRL-based cooperative edge caching framework among BSs, for coping with the challenge of offloading duplicated traffic and improving the specific QoS. Different from other caching strategies, the proposed framework is self-adaptive in the dynamic environments aiming to reduce the access delay of users as well as the traffic burden on the backhual. Finally, compared with LRU, LFU, FIFO and Oracle, trace-driven simulation results have shown that the proposed framework can achieve excellent performance in terms of the average delay, the hit rate and the traffic offload of backhaul.

ACKNOWLEDGMENT

The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions. This work is partially supported by the National Key R&D Program of China (2018YFC0809803), China NSFC (Youth) through grant 61702364, China NSFC GD Joint fund U1701263 and China NSFC through grant 61872266.

REFERENCES

[1] X. Wang, M. Chen, Z. Han, et al., "TOSS: Traffic offloading by social network service-based opportunistic sharing in mobile social networks," *in Proc. IEEE INFOCOM*, Apr.-May 2014, pp. 2346-2354.
[2] X. Li, X. Wang, K. Li, Z. Han, et al., "Collaborative multi-tier caching in heterogeneous networks: modeling, analysis, and design," *IEEE Trans. Wireless Commun.*, vol. 16, no. 10, pp. 6926-6936, Oct. 2017.
[3] X. Wang, M. Chen, V. C. Leung, Z. Han and K. Hwang, "Integrating Social Networks with Mobile Device-to-Device Services," *in Proc. IEEE Transactions on Services Computing*,2018:1-1.
[4] X. Wang, M. Chen, T. Taleb, et al. "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] M. Sheng, C. Xu, J. Liu, J. Song, X. Ma, and J. Li, "Enhancement for content delivery with proximity communications in caching enabled wireless networks: Architecture and challenges," *IEEE Commun. Mag.*,vol. 54, no. 8, pp. 70-76, Aug. 2016.
[6] E. Zeydan, et al., "Big data caching for networking: Moving from cloud to edge," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 36-42, Sep. 2016.
[7] N. Golrezaei, et al., "FemtoCaching: Wireless video content delivery through distributed caching helpers," *in Proc. IEEE INFOCOM*, Mar. 2012, pp. 1107-1115.
[8] N. Golrezaei, et al., "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402-8413, Dec. 2013.
[9] S. Woo, E. Jeong, S. Park, J. Lee, S. Ihm, and K. Park, "Comparison of caching strategies in modern cellular backhaul networks," *in Proc. ACM MobiSys*, Jun. 2013, pp. 319-332.
[10] X. Li, X. Wang, Xiao S, and V. C. M. Leung, "Delay performance analysis of cooperative cell caching in future mobile networks," *in Proc. IEEE ICC*, May 2015, pp. 5652-5657.
[11] S. H. Chae, J. Y. Ryu, T. Q. S. Quek, and W. Choi, "Cooperative transmission via caching helpers," *in Proc. IEEE GLOBECOM*, Dec. 2015, pp. 1-6.
[12] J. Gu, W. Wang, A. Huang, et al., "Distributed cache replacement for caching-enable base stations in cellular networks," *in Proc. IEEE ICC*, Jun. 2014, pp. 2648-2653.
[13] P. Blasco and D. Gunduz, "Learning-based optimization of cache content in a small cell base station," *in Proc. IEEE ICC*, Jun. 2014, pp. 1897-1903.
[14] C. Zhong, M. C. Gursoy, S. Velipasalar, "A deep reinforcement learning-based framework for content caching," *in Proc. CISS*, Mar. 2018, pp. 1-6.
[15] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE Trans. Netw.*, vol. 24, no. 5, pp. 2795-2808, Oct. 2016.
[16] 3GPP, "Further advancements for E-UTRA physical layer aspects (release 9)," TR 36.814 V1.2.0, Jun. 2009.
[17] H. V. Hasselt, A. Guez, D. Silver, "Deep Reinforcement Learning with Double Q-learning," *in Proc. AAAI*, Feb. 2016, pp. 2094-2100.
[18] X. Li, X. Wang, P.-J. Wan, Z. Han, and V. C. M. Leung, "Hierarchical edge caching in device-to-device aided mobile networks: modeling, optimization, and design," *IEEE J. Sel. Areas Commun.*, 2018.
[19] C. Wang, S. Wang, D. Li, X. Wang, X. Li, and V. C. M. Leung, "Q-Learning based Edge Caching Optimization for D2D Enabled Hierarchical Wireless Networks," *IEEE MASS*, Chengdu, China, 2019, pp. 55-63.
[20] S. Müller, O. Atan, M. V. D. Schaar, et al., "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 1024-1036, Feb. 2017.