Edge Caching Replacement Optimization for D2D Wireless Networks via Weighted Distributed DQN

Ruibin Li[1], Yiwei Zhao[1], Chenyang Wang[1], Xiaofei Wang[1], Victor C. M. Leung[2,3], Xiuhua Li[4], Tarik Taleb[5]
[1]College of Intelligence and Computing, Tianjin University, Tianjin, China
[2] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China
[3]Dept. Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada
[4]College of Intelligence and Computing, Chongqing University, Chongqing, China
[5]School of Electrical Engineering, Aalto University, Helsinki, Finland
Email: {leeruibin, yiweizhao, chenyangwang, xiaofeiwang}@tju.edu.cn
vleung@ece.ubc.ca, lixiuhua1988@gmail.com, tarik.taleb@aalto.fi

***Abstract*—Duplicated download has been a big problem that affects the users' quality of service/experience (QoS/QoE) of current mobile networks. Edge caching and Device-to-Device communication are two promising technologies to release the pressure of repeated traffic downloading from the cloud. There are many researches about the edge caching policy. However, these researches have some limitations in the real scenarios. Traditional methods are lacking the self-adaptive ability in the dynamic environment and privacy issues will occur in centralized learning methods. In this paper, based on the virtue of Deep Q-Network (DQN), we propose a weighted distributed DQN model (WDDQN) to solve the cache replacement problem. Our model enables collaboratively to learn a shared predictive model. Trace-driven simulation results show that our proposed model outperforms some classical and state-of-the-art schemes.**

I. Introduction

At present, the pervasive utilization of the Internet of Things (IoT) technology has led to the rocket-increasing requirements for existing cellular networks, which results in the upsurge of mobile traffic. In fact, duplicated download [1] is one of the desirable problems that affect the users' quality of service (QoS). Mobile Edge Caching (MEC) is a promising technology to release the pressure of repeated traffic downloading for network operators. Illustrated as Fig. 1, contents can be cached in proximity to the edge of networks (e.g. base stations and mobile devices), which shortens the transmission delay and reduces the traffic pressure of backhaul networks by device-to-device (D2D) communications as well.

Recently, the research on edge caching has been more specific and many efforts on network resources (user mobility, content popularity, spectrum usage, etc.) optimization have been explored. Authors in [2]- [4] proposed the edge caching strategies by optimizing the single factor, such as user mobility, file utility of system and multi-user participation. Others considered the joint optimization, Peiyan et al. [5] addressed the tradeoff between transmission distance and network capacity. In [6] [7], authors showed that combined factors such as file popularity, user preferences and mobility may contribute the better network performance. However, most of the related studies are based on the assumptions of small-scale scenarios or the homogeneous distribution of content popularity.

Furthermore, edge caching strategies based on conventional optimization approaches are lacking self-adaptive ability in dynamic environment. To tackle this problem, some studies explored the learning-based methods (e.g. machine learning, deep learning, etc.) to design the edge caching strategies. Pang et al. [8] showed the advantages of long short-term memory (LSTM) in terms of time series prediction to realize the cooperative edge caching. Authors in [9] attempted the multi-agent reinforcement learning (RL) to achieve the long-term reward of the cache replacement algorithm. Deep Q-learning Network (DQN) solves the problem of action/state space-explosion in RL, which was used to settle the problem of joint edge computing and caching resource allocation [10]. However, most of the studies are centralized training models, which may lead to a huge burden on the datacenter and pose serious security risks to users' privacy.

Motivated by the aforementioned, we propose a weighted distribute DQN training architecture. In our architecture, every base station (BS) trains its own DQN model based on its local data and local content popularity. The cloud server will aggregate all the local model parameters in a reward-based adaptive boosting manner and then disperse the global model. The distributed training method can reduce the privacy risk of users and the reward-based aggregation manner improves the BS's self-adaptive ability in the dynamic environment. The contributions of this paper are summarized as follows:

- This paper considers the difference of content popularity, user preferences and social networks in different BSs and establishes a D2D sharing model. After that, we propose a replacement policy for the edge cache content.
- We model the cache content replacement problem as a Markov decision process (MDP) and train the DQN model via a reward-based distributed manner which reduces the privacy risk of users, speeds up the training process and improves the BS's self-adaptive ability in the dynamic environment.
- Experimental results show that proposed policy achieves

better performance compared to the algorithms including FIFO, LRU, LFU and Centralized DQN, the hit rate enhancements are 21%, 23%, 18% and 5%, and the offload rate enhancements are 15%, 18%, 12% and 5%.

The rest of this paper is organized as follows. We introduce the system model in Sec. II. The details of the edge caching policy are presented in Sec. III. We use the weighted distributed DQN model to solve the problem at BS since the massive state space in Sec. IV. We conduct the large-scale real trace-based experiment in Sec. V. Finally, we conclude the paper in Sec. VI.

II. System Model

A. Basic Definition

Shown as Fig. 1, we divide the architecture into three layers, the cloud layer is in charge of the parameters aggregation computation using a cloud server. There are B BSs in the edge layer, denoted as $\mathcal{B} = \{b_1, b_2, ..., b_B\}$, with the cache size of $C_{\mathcal{B}} = \{c_1^{BS}, c_2^{BS}, ..., c_B^{BS}\}$. Mobile users with the local cache buffer c^U are uniformly distributed in the coverage of each BS, represented as $S_B^{BS} = \{U_1^{BS}, U_2^{BS}, ..., U_B^{BS}\}$ where U_i^{BS} denotes the set of users in the user layer. Assuming that there are $\mathcal{F} = \{f_1, f_2, ..., f_F\}$ files stored in the content library. For each BS i, it stores local associated users' cache state matrix $\Omega_i = (x_{u,f})^{U_i^{BS} \times F}$, where $x_{u,f} = 1$ means user u has content f, otherwise $x_{u,f} = 0$.

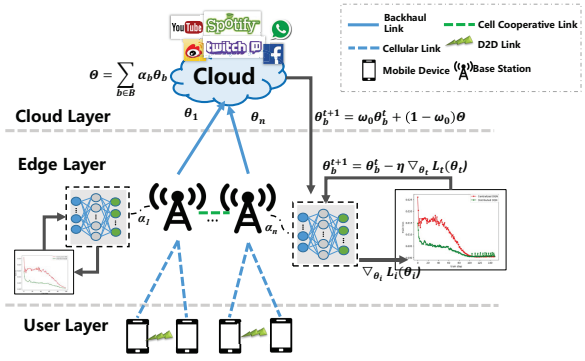


Fig. 1. Illustration of system architecture.

B. Content Popularity and User Preferences

1) Content Popularity: $\mathcal{F}$ denotes the probability distribution of content requests in the content library. Generally, the content popularity follows the MZips [11] distribution as:

$$\omega_f^b = \frac{K}{(f_i + q)^\alpha}, \tag{1}$$

where f_i is the local popularity index of content f in the descending order, α is the skewness factor and $q \geq 0$ defines the plateau shape near to the left-most part of the distribution, K is the sum of $1/(f_i + q)^\alpha$.

2) User Preferences: denoted as p_u^f, is the probability distribution of a users request for each content. Content popularity and user preference are assumed to be stochastic.

C. D2D Sharing Model

In order to model the D2D sharing activities, physical domain and social domain are two important factors should be concerned.

1) Physical Domain: Due to the physical constraints such as signal attenuation, only a subset of nodes that are closely enough (e.g., with the detectable signal strength) can be feasible relay candidates for the node n. Thus we introduce the physical graph $\mathcal{G}_P^b \triangleq \{U_b^{BS}, \mathcal{E}_P^b\}$ where the set of nodes associated with BS b (U_b^{BS}) is the vertex set and $\mathcal{E}_P^b \triangleq \{(n,m) : e_{nm}^p = 1, \forall n, m \in U_b^{BS}\}$ is the edge set where $e_{nm}^p = 1$ if and only if node m is a feasible relay for node n.

2) Social Domain: The social graph $\mathcal{G}_S^b$ has the same form as the physical graph. Here, the edge set is given as $\mathcal{E}_S^b \triangleq \{(n,m) : e_{nm}^s = 1, \forall n, m \in U_b^{BS}\}$. The edge $e_{nm}^s = 1$ in the set $\mathcal{E}_S^b$ is mainly based on the social network research that one would establish a D2D share communication with another mainly by two key social phenomena: social trust [12] and social reciprocity [13].

3) Communication Graph: Based on the Physical Graph and the Social Graph, we can get the communication graph $\mathcal{G}_C^b \triangleq \{U_b^{BS}, \mathcal{E}_C^b\}$, where $\mathcal{E}_C^b \triangleq \{(n,m) : e_{nm}^c = 1, \forall n, m \in U_b^{BS}\}$ is the edge set where $e_{nm}^c = 1$ if and only if there is a path (the length of which is less than a threshold l) between n and m in the Physical Graph as well as the Social Graph.

4) D2D Share Probability: Considering the request ID of user u at time t as statistics variable X and the shared content ID by the user v at time t as statistics variable Y. If $X = Y$ then there may be a D2D communication between u and v. So there is some correlation between X and Y. We use the correlation coefficient r_{uv} during a certain time to measure the D2D share probability between u and v. It is given by:

$$r_{uv} = \frac{Cov(X,Y)}{\sqrt{Var[X]Var[Y]}}, \tag{2}$$

where $Cov(X,Y)$ denotes the covariance between X and Y, $Var[X]$ means the variance of X. we use the correlation coefficient to describe the D2D share probability between user pair (u,v). We define $r_{uu} = 0$ for the pair (u,u). After getting the $\mathcal{R} = (r_{uv})^{1 \times U_b^{BS}}$, we normalize $\mathcal{R}$ to make sure $\sum_1^B \mathcal{R} = 1$.

D. Content Transmission between BS and User

If the content request from the mobile user u cannot be satisfied through D2D sharing, then users have to ask BS for the content. The *BS Serving Rate* P_u^{BS} can be obtained as $P_u^{BS} + \sum_{v \in U_b^{BS}} r_{uv} e_{uv}^c = 1, \forall u \in U_b^{BS}$.

III. Caching Replacement Policy Optimization

We are aiming to optimize the network edge caching policy by offloading the contents to mobile users via D2D communication, and the system cost of content exchange between BSs and cloud will be reduced as well.

A. D2D Caching Gain

To measure the total D2D sharing in the local BS, we define Ga_b^{D2D} which is calculated via the D2D sharing probability, content size and user cache state.

For a local BS b and all the pairs (u, v) of its associated users U_b^{BS}, v can get the requested content f from u, if u has the content ($x_{u,f} = 1$) and v does not ($x_{v,f} = 0$) with the probability $r_{vu}e_{vu}^c$ during time slot t. Thus the content offload gain via D2D communication between u and v at time slot t can be obtained as $f_{v,t}r_{vu}e_{vu}^c$ where $f_{v,t}$ represents the content size of file that user v requests at time slot t. Then the total D2D gain in the BS b at time slot t can be calculated as:

$$Ga_{b,t}^{D2D} = \sum_{u,v \in U_b^{BS}} f_{v,t}r_{vu}e_{vu}^c x_{u,f}(1 - x_{v,f}). \tag{3}$$

It gives us global insight into the D2D sharing activities, which can be regarded as a reference for the system efficiency.

B. Backhaul Traffic Cost

If the request can not be satisfied by the D2D sharing, u can get the content if the content is cached in the local BS. Otherwise, the local BS has to download the content from other BS or the core network, we regard these requests as the cost of the system since these requests increase the burden of the backbone network.

We define the backhaul traffic cost at time slot t as $Co_{b,t}^{D2D}$, through the communication probability between user u and BS b, content size and BS cache state. It can be gotten from

$$Co_{b,t}^{D2D} = \sum_{u \in U_b^{BS}} f_{u,t}P_u^{BS}(1 - x_b^{BS}). \tag{4}$$

It gives us insight into the backhaul traffic that can not be avoided.

C. System Reward

Based on the above reference in our system, we define $R_{b,t}(S)$ as the total reference of the system performance at time slot t. The system reward of BS b can be calculated as

$$R_{b,t}(S) = \beta_0 Ga_{b,t}^{D2D} - \beta_1 Co_{b,t}^{D2D}, \tag{5}$$

where $\beta_0, \beta_1 \in (0, 1)$. They are users' preferences for D2D communication and BS communication.

Note that β_0, β_1 will be different in the different BSs, since every group has their own preference for D2D and BS communication considering the difference in time cost and convenience of these two kinds of communication modes.

The system reward $R_{b,t}(S)$ is essential for the following DQN training, it directly decides which content should be replaced in the system.

D. Caching Replacement Policy

Once the new requests come, the local BS will determine how to update the cache state of its associated users and which content in the user's cache should be replaced. Therefore the optimization problem can be defined as:

$$\begin{aligned} &\max_{A^*(f^-, f^+)} \sum_{b \in \mathcal{B}} R_{b,t}(S^{'}|S), \\ &s.t. \sum_{i \in \mathcal{F}} x_{u,i}f_i \le c_u^U, \forall u \in U_b^{BS} \\ &\quad \sum_{i \in \mathcal{F}} x_{b,f}f_i \le c_b^{BS} \\ &\quad x_{u,f}, x_{b,f} \in \{0, 1\} \end{aligned} \quad , \tag{6}$$

where S denotes all the local BS state before the action A^*, and $S^{'}$ denotes the new BS state after action A^*. Our aim is to optimize the best action A^* at current state to maximize the total system reward (maximize the D2D caching gain and minimize the backhaul traffic cost) of the next state while satisfying the all the constraints about the cache size of all mobile users and BSs.

Problem (6) has the same structure with the problem formulated in [14], which has been proved as NP-hard. It is difficult to find a mathematically optimal solution. Instead, we use the DQN model to get the approximate optimal solution in the next section.

IV. Weighted Distributed DQN based cache replacement policy

We model the cache replacement problem as a Markov Decision Process (MDP) problem and use a weighted distributed DQN model to solve the problem. There are three main phases during the training which are as follows:

A. DQN Model Training

The cache state of BS b is set as S_0^b initially. Every time slot the DQN agent would observe current D2D cache state S_t^b and choose a replacement action A_t to maximize the possible system D2D traffic.

The reward function is the most important part of DQN, since it directly decides which content should be replaced. In our system, the difference between the total reward of old state and new state $r_t = (R_{b,t}^{S^{'}} - R_{b,t-1}^{S})$ is regarded as the reward of the action. For a standard reinforcement learning method, the sequence $\{S_0^b, A_0, S_1^b, ..., A_{t-1}, S_t\}$ could be modeled as the MDP directly. The discounted reward for the future R_t could be gotten through:

$$R_t = \sum_{t'}^{T} \gamma^{t'-t} r_t, \tag{7}$$

where $\gamma \in (0, 1]$. The target of the DQN is to maximize the action-value function:

$$Q^*(S, A) = \max_{\pi} \mathbb{E}[R_0|S_0 = S, A_0 = A, \pi], \tag{8}$$

where π is the strategy for choosing of the best action. According to the Bellman equation, it is equal to maximize the expected value of $r + \gamma Q^*(S^{'}, A^{'})$, if the optimal value $Q^*(S^{'}, A^{'})$ of the sequence at the next time step is known.

$$Q^*(S, A) = \mathbb{E}[r + \gamma \max_{A^{'}} Q^*(S^{'}, A^{'})|S, A]. \tag{9}$$

We use a neural network function approximation with weights θ to estimate $Q^*, Q(S, A, \theta) \approx Q^*(S, A)$. The loss function is defined as:

$$L_i(\theta_i) = \mathbb{E}[(y_i - Q(S, A; \theta_i))^2], \tag{10}$$

y_i is the target, which is calculated by the previous iteration parameter result θ_{i-1}. The gradient of the loss function is:

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}[(y_i - Q(s, a; \theta_i)) \nabla_{\theta_i} Q(S, A; \theta_i)], \tag{11}$$

the process is shown in Algorithm 1. Where $\mathcal{M}$ represents the whole experience pool and $\widetilde{\mathcal{M}}$ is a random sample of $\mathcal{M}$. Throughout the training process, we use two Q-network which are evaluation network Q_e and target network Q_t. Q_t is mainly used to give cache replacement results while Q_e is mainly used to update the parameters based on the new transition T. We will replace Q_t with the parameters of Q_e every ϕ times.

Algorithm 1 WDDQNLearning

Initialize Local BS Parameter and eporch K
for $i \in [0, K]$ **do**
  R_i = *new request*
  Get init State S
  if R_i *in* $cache_u$ **then**
    A_i = *Get the Content from* $cache_u$
  else if R_i *in cache of* $neighbor_u$ **then**
    A_i = *Get Content from neighbors*
  else
    A_i = *Get content from BS*
  end if
  Get Reward r_i
  Get current State S'
  Store $T = (R_i, S, S', A_i, r_i)$ *into* $\mathcal{M}$
  update $Q_e(S, A; \theta_i)$ *with* $\widetilde{\mathcal{M}}$ *and* $\nabla_{\theta_i} L_i(\theta_i)$
  update Q_t *every* ϕ *times*
end for

B. Global Model Aggregation

In the aggregation phase, each BS disperses its parameters to the cloud server after M epoch and the Cloud will integrate these parameters. The process is shown as Fig. 2.

The aggregation frequency M is set as [15] to make the available resource is most efficiently used. The loss in supervised learning is related to the training data and it can be regarded as the reward of the learning process. However, in the DQN model, the loss is the difference between current and previous Q-network which has no direct relation with the training data but shows the convergence of the Q-network.

Inspired by [16], we use a reward-based adaptive boosting manner in the supervised learning problem to learn the BS neural network and the aggregation process. The reward has a strong relation with the input transition (R_i, S, S', A_i, r_i). We employ the average reward $\overline{r_b}$ of BS b in a certain time to represent the BS's contribution to the global model. The

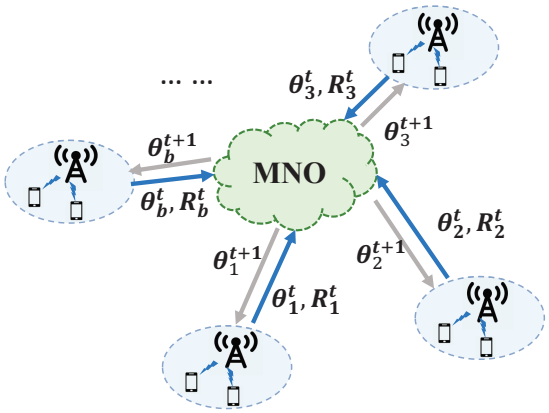


Fig. 2. Global Parameter Aggregation Process

contribution rate α_b of the BS b is calculated by:

$$\alpha_b = \overline{r_b} \Big/ \sum_{i \in \mathcal{B}} \overline{r_i}. \tag{12}$$

The α_b is regarded as an important measurement for local BS's contribution. Thus, the guidance parameter Θ of the global model can be calculated by:

$$\Theta = \sum_{b \in \mathcal{B}} \alpha_b \theta_b. \tag{13}$$

C. Local Model Update

To make the BS agent more adaptive and make full use of the training results of other BSs, we would update the local model via the global model after the aggregation.

Algorithm 2 Local Parameter Update

Initialize episode K and aggregation frequency M.
for *episode* $t \in [0, K]$ **do**
  for *each BS* b **do**
    if $t\%M == 0$ **then**
      $\theta_b^{t+1} = \omega_0 \theta_b^t + (1 - \omega_0)\Theta$
    else
      $\theta_b^{t+1} = \theta_b^t - \eta \nabla_{\theta_t} L_t(\theta_t)$
    end if
  end for
end for

The local update process is shown in algorithm 2. If there no aggregation happens, local DQN will update the parameter normally by gradient descent. Otherwise, we update the model by global Θ, local θ and ω_0. ω_0 is the factor that controls the proportion of Θ, θ. The update process can be calculated as:

$$\theta_b^{new} = \omega_0 \theta_b^{old} + (1 - \omega_0)\Theta \tag{14}$$

The computation complexity mainly includes collecting transitions and back propagation. Assume that the length of replay memory is M, the complexity is $O(M)$. Let a and b denote the layer number and the number of units in each layer. The complexity of training parameters with back

propagation and gradient descent requires $O(mabk)$, where m is the number of transitions randomly sampled from the replay memory and k denotes the number of iterations, respectively. Specifically, the replay memory stores M transitions which requires the space complexity of $O(M)$ and it requires the space complexity of $O(ab)$ to deal with the storage complexity by the parameters of DQN.

V. Experiment

In this section, we evaluate the proposed cache replacement policy and give the experiment results.

A. Parameter Settings

Initially, the local content popularity via MZip's law with $\alpha = 0.8$. Further, we set different q in Eq. (1) to set different local content popularity, shown as Fig.3. Other parameters are shown in the Tab.I.

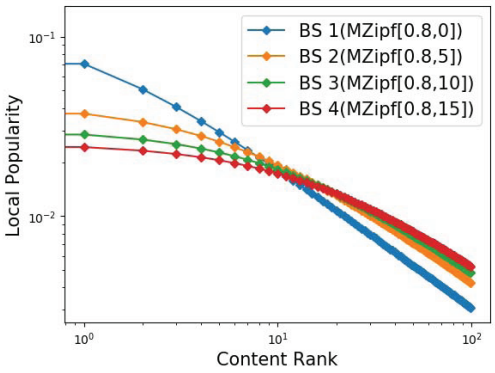


Fig. 3. Different local content popularity

TABLE I
Experiment Setting

Content Number	100
Users Number	50
User Cache Size	10
BS Cache Size	100
Q_t update frequency ϕ	250
Local Users Cache State	$(0)^{U\times F}$
BS Cache State	$(0)^{1\times F}$
Local User Preference	$(f_{uf})^{U\times F}$
Local U2U Share Probability	$(p_{uv})^{U\times U}$
Local U2B Share Probability	$(p_{ub})^{U\times B}$

B. Experimental Results

We compare the training loss of Centralized and Distributed DQN. Fig. 4 shows that the distributed DQN model converges faster than the Centralized DQN model and gets less training loss which means the distributed DQN model gets a more stable Q-network. Even though there are some concussions after the model converged due to the local model update, the model can be stabilized very quickly soon after that.

We also evaluate the performance of the strategy in terms of **hit rate** and **offload rate**.

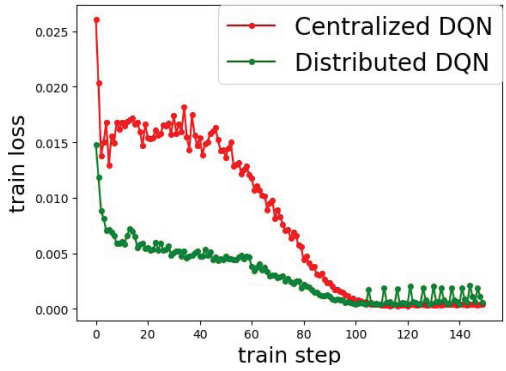


Fig. 4. Loss Comparison Between Centralized and Distributed DQN

Hit Rate: denoted as $HR = N_h/N_r$, where N_h is the number of satisfied user requests, N_r is the total number of requests.

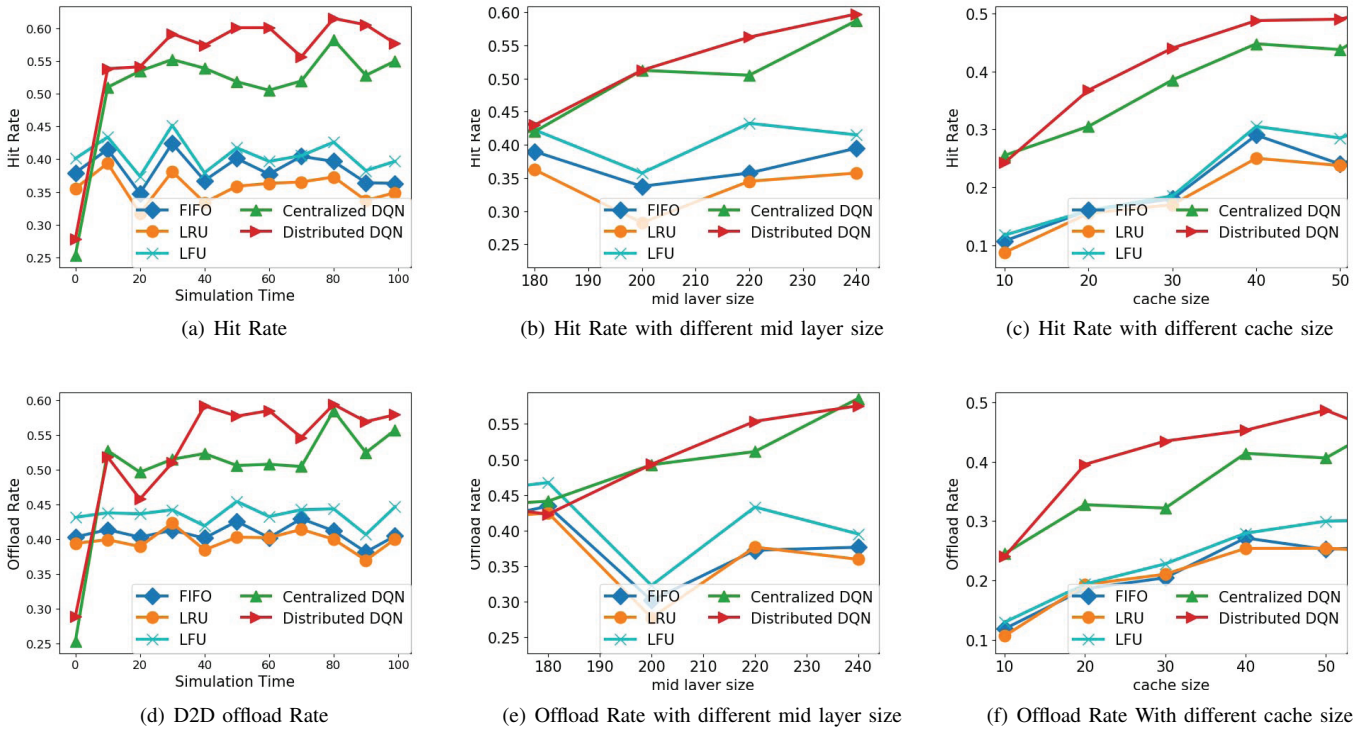
Offload Rate: we use the offload rate to compare the D2D offload traffic with the traffic of the total requests. T_{total} denotes the total requests traffic during the time slot and T_{d2d} is the D2D offload traffic. It can be calculated as $OR = T_{d2d}/T_{total}$.

Compared with state-of-the-art algorithms such as FIFO, LRU, LFU, and Centralized DQN, we can see that the proposed Distributed DQN outperforms others, shown as Fig.5. The users' requests changed With the simulation time moving so that the metrics values have a little concussion. We can observe the proposed distributed DQN model achieves the almost 58% average hit rate performance, which is better than 37% (FIFO), 35% (LRU), 40% (LFU), 53% (Centralized DQN). The Offload Rate of Distributed DQN is also better as we can observe in Fig.5(d).

To better explore the network performance of the proposed algorithm, we compare the performance of WDDQN with different cache sizes and mid layer sizes. Shown in the Fig.5, the proposed WDDQN model increases the performance of 21% (FIFO), 23% (LRU), 18% (LFU), 5% (Centralized DQN) and 15% (FIFO), 18% (LRU), 12% (LFU), 5% (Centralized DQN) in terms of hit rate and offload rate, repectively. The hit rate and the offload rate are always better compared with other algorithms as the cache size increases. Meanwhile, the performance of our proposed model will get better and better as the mid layer size increases since it can greatly adapt to the local requests of the users. Moreover, the worst of our models can also achieve the effect of the centralized DQN model and guarantee the user's privacy through distributed ways at the same time.

VI. Conclusions

In this paper, we propose a weighted distributed DQN based edge caching strategy to tackle the duplicated downloading problem in wireless networks. Specifically, we use MDP to model the edge caching process and communication graph which considers both physical and social domains to model



(a) Hit Rate (b) Hit Rate with different mid layer size (c) Hit Rate with different cache size

(d) D2D offload Rate (e) Offload Rate with different mid layer size (f) Offload Rate With different cache size

Fig. 5. Performance Demonstration of Hit Rate and Offload Rate with different Algorithm

the D2D sharing process. To find the optimal replacement policy, we propose weighted distributed DQN architecture. The "weighted" in the architecture means that in the aggregation process we give each BS a weight based its average reward and do the global model aggregation via weighted summation. During this process, D2D Caching Gain and Backhaul Traffic Cost are two important indicators which are utilized to quantify the reward. We alleviate the whole training efficiency by the proposed architecture, which allows the training data remained on the local BS. Simulation results show the proposed strategy can outperform the FIFO, LRU, LFU and Centralized DQN schemes in terms of request hit rate and offload rate, and it also has a faster convergence speed compared with the Centralized DQN model.

ACKNOWLEDGMENT

This work is partially supported by the National Key R&D Program of China (2019YFB2101901, 2018YFC0809803), and China NSFC (Youth) through grant 61702364.

REFERENCES

[1] Wang, Xiaofei, et al. "TOSS: Traffic offloading by social network service-based opportunistic sharing in mobile social networks." IEEE INFOCOM 2014-IEEE Conference on Computer Communications. IEEE, 2014.

[2] Chen, Min, et al. "Green and mobility-aware caching in 5G networks." IEEE Transactions on Wireless Communications 16.12 (2017): 8347-8361.

[3] Chen, Xu, et al. "Efficient multi-user computation offloading for mobile-edge cloud computing." IEEE/ACM Transactions on Networking 24.5 (2015): 2795-2808.

[4] Xiao, Falu, et al. "Max-FUS Caching Replacement Algorithm for Edge Computing." 2018 24th Asia-Pacific Conference on Communications (APCC). IEEE, 2018.

[5] Yuan, Peiyan, et al. "Collaboration Improves the Capacity of Mobile Edge Computing." IEEE Internet of Things Journal (2019).

[6] Zhang, Shan, et al. "Cooperative edge caching in user-centric clustered mobile networks." IEEE Transactions on Mobile Computing 17.8 (2017): 1791-1805.

[7] Liu, Wei, et al. "Mobility-Aware Video Prefetch Caching and Replacement Strategies in Mobile-Edge Computing Networks." 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 2018.

[8] Pang, Haitian, et al. "Toward smart and cooperative edge caching for 5g networks: A deep learning based approach." 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS). IEEE, 2018.

[9] Jiang, Wei, et al. "Multi-Agent Reinforcement Learning Based Cooperative Content Caching for Mobile Edge Networks." IEEE Access 7 (2019): 61856-61867.

[10] Xu, Fangmin, et al. "DQN Inspired Joint Computing and Caching Resource Allocation Approach for Software Defined Information-Centric Internet of Things Network." IEEE Access 7 (2019): 61987-61996.

[11] Hefeeda, Mohamed, and Osama Saleh. "Traffic modeling and proportional partial caching for peer-to-peer systems." IEEE/ACM Transactions on networking 16.6 (2008): 1447-1460.

[12] Govier, Trudy. Social trust and human communities. McGill-Queen's Press-MQUP, 1997.

[13] Gintis, Herbert. "Strong reciprocity and human sociality." Journal of Theoretical Biology 206.2 (2000): 169-179.

[14] Leiserson, Charles Eric, et al. Introduction to algorithms. Vol. 6. Cambridge, MA: MIT press, 2001.

[15] Wang, Shiqiang, et al. "When edge meets learning: Adaptive control for resource-constrained distributed machine learning." IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE, 2018.

[16] Lv X, Guan Y, Deng B. "Transfer learning based clinical concept extraction on data from multiple sources." Journal of Biomedical Informatics, 2014, 52:55-64.