Cooperative Hybrid-Caching for Long-Tail Distribution Request with Deep Reinforcement Learning

Weibao He[1(✉)], Fasheng Zhou[2], and Dong Tang[2]

[1] School of Physics and Materials Science, Guangzhou University, Guangzhou 510006, China
hwb@e.gzhu.edu.cn

[2] School of Electronics and Communication Engineering, Guangzhou University, Guangzhou 510006, China
{zhoufs,tangdong}@gzhu.edu.cn

Abstract. Wireless caching is regarded for alleviating network congestion in next-generation communications. In this work, we focus on the impact of input data non-uniformity on neural network training when using deep learning to solve the wireless cache strategy. In particular, in wireless caching, the assumed user request is generally a classical long-tailed distribution: Zipf law. We address this problem from cache models and deep reinforcement learning models. On the one hand, base station will prefetch partial most popular contents on the user side to reduce the partiality of caching strategies. On the other hand, the trick of deep long-tail learning is added to prevent the neural network from been over-fitting caused by inputs are concentrated in the most popular files. The performance for different reinforcement learning methods is analyzed, which show that our method can achieve better performance and latency than the existing content caching.

Keywords: Edge caching · Deep reinforcement learning · Deep long-tail learning

1 Introduction

Wireless network devices and traffic will expand exponentially over the next generation of mobile communications. Specifically, the proportion of streaming media in cellular network traffic is growing rapidly [1]. Wireless caching is promised as a technique to improve network resource utilization because it can cache popular contents with off-peak traffic and make cache strategy. When users ask for popular content, the base station(BS) sends it directly via the wireless link, which reduces the peak connecting traffic. What's more, the neighboring BS may share the cached content via wired or wireless links to expand the service's capacity and coverage.

K. Rabie et al. (Eds.): IoTaaS 2022, LNICST 506, pp. 148–160, 2023.
https://doi.org/10.1007/978-3-031-37139-4_14

In general, caching strategy depends not only on the content popularity, but also on the BS-user channel, BS capacity, wireless cooperative network topology, etc. At the same time, the content of the Internet changes rapidly, and it is necessary to redesign the caching strategy between BSs periodically to suit the environment. Deep reinforcement learning (DRL) has strong adaptability to the environment, and it will be effective to solve the wireless cache strategy [2]. In [3], the authors use long short term memory (LSTM) and supervised deep deterministic policy gradient (SDDPG) to learn the caching strategy based on time-variant content popularity.

However, since the distribution of popular contents is non-uniform, taking user requests as observations may make the neural network too dependent on the most popular contents [4]. Referring to Deep long-tail learning for visual recognition [5], commonly used solutions include meta reinforcement learning [6], weight balance [7], etc. However, not all methods can be applied to DRL. Motivated by this, we strive to balance the contribution between the most popular content and the less popular content, which includes the innovations in cache model and proposed DRL method. In this work, we reconsider the reward, state and action design in DRL and based on the deep deterministic policy gradient [8] (DDPG) algorithm, transform anti-longtail techniques to obtain the proposed method. The main components of our contributions includes:

1) Considering a hybrid-caching model, which the BS can prefetch the most popular contents to users. In this way, wireless caching can further reduce the delay caused by the most popular contents and cut down the weight of the most popular contents.
2) A DDPG-based DRL method is proposed to solve the formulated problem, which can achieve advanced low-latency performance compared to tradition DRL method.
3) For popular streaming media data, the user's request delay for the most popular contents will be zero, and the dependence of the BS on the most popular contents in proposed method will be reduced.

Notations: $A >> B$ denotes that A is far greater than B; while $<<$ is far smaller than. $\mathbb{E}$ is the expectation operator. ∇_θ denotes the gradient with respect to θ, and $A|B$ denotes the event A conditioned on B.

2 System Model

2.1 Wireless Model

The considered centralized wireless hybrid-caching model contains M base stations(BSs) and U users, denoted by $\Phi_M = \{1, 2, \ldots, M\}$ and $\Phi_U = \{1, 2, \ldots, U\}$, respectively. The service distance of each BS is limited within l_p. Assuming that the distance between BS m and user u is l_{mu}. Therefore, the user served by m-th BS is denoted by U_m, which the set is defined by $\Phi_U^{(m)} \triangleq \{l_{um} < l_p, u \in \Phi_U\}$. Consider the achievable radio data rate between BS m and user u with transmission bandwidth B is denoted by:

$$R_{mu} = B\log(1 + \frac{P_t l_{mu}^{-\alpha}}{\sigma^2}), \tag{1}$$

where P_t is transmit power, α is path-loss factor and σ^2 is the power of noise. For BSs cooperative cache scenario, assume that the BSs cooperate over a wireless link. The achievable radio data rate with bandwidth of B_{co} and the distance between neighboring BSs l_{co} is denoted by

$$R_{co} = B_{co}\log(1 + \frac{P_t l_{co}^{-\alpha}}{\sigma^2}). \tag{2}$$

Assuming that users are randomly spread across the service area of the BS, and the distance between neighboring BSs will not change. Depending on the different distances between users, the BS should adopt a reasonable cache strategy to minimize the transmission delay.

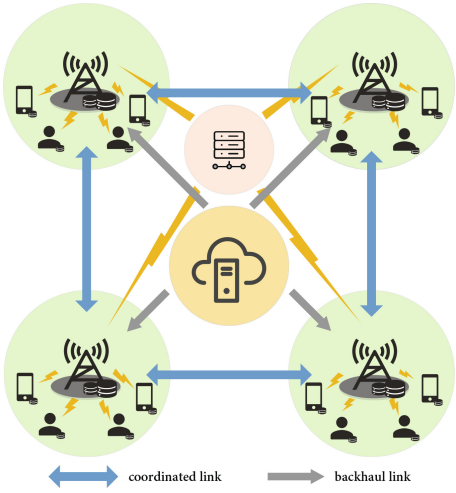


Fig. 1. A centralized coordinated hybrid-caching network.

2.2 Caching Model

As shown in Fig. 1, we show the centralized wireless hybrid-caching model. Assuming that BSs can collaborate to serve users, meanwhile users also have extremely limited cache space for BS prefetch popular contents to reduce delay. The user cache can effectively reduce the rely of the learning algorithm to the most popular contents. The capacity of BS and user m are denoted by C_{BS} and C_u. In particular, there is a central control unit (CU) in the model, which estimate the caching strategies of each BS through global channel information and user requests. Assuming that the total number of content and the set of content are denoted by F and $\mathcal{F}$. To

simplify analysis, assume that each content size is Z bits. Practically, the capacity of BS is generally smaller than the content number and the capacity of user is much smaller than BS *i.e.,* $F > C_{BS} \gg C_u \geq 0$.

Since user cache capacity is extremely limited, we divide a content into K parts and the user only cache the first part of each content. Assuming that the physical capacity of user is C_{user}, then, the capacity of user is denoted by $C_u = KC_{real}$. In this way, on the one hand, the partial cache will expanded the cache range. On the other hand, the stored contents can also reduce the user delay to 0 for streaming video or audio.

The BS cache strategy in time slot t in denoted by $\mathbb{A}^t = \{0,1\}^{M\times F}$, with each of element $a^t_{m,f} = 1$ indicating content f is cached in BS m in time slot t if $a^t_{m,f} = 1$ or otherwise not if $a^t_{m,f} = 0$. The user request in time slot t can be similarly defined by $\mathbb{B}^t = \{0,1\}^{U\times F}$ where each element $b^t_{u,f} = 1$ or 0 indicating that the g-th content is requested by user u in time slot t or not. The user cache in time slot t can be similarly defined by $\mathbb{G}^t = \{0,1\}^{U\times F}$ where each element $g^t_{u,f} = 1$ or 0 indicating that the g-th content cached by user u in time slot t or not. The contents are sorted in descending order by their popularity, and obey the classic long-tailed distribution : Zipf distribution, which can be expressed by:

$$P_i = \frac{i^{-\gamma}}{\sum_{j=1}^{F} j^{-\gamma}} \forall i \in \mathcal{F}, \tag{3}$$

where γ is Zipf factor. The larger γ is, the more request probability will concentrate on the most popular contents.

When contents are cached in a nearby BS instead of a service BS, the service BS will fetch the contents through a cooperative BS-BS link. Therefore, the CU can jointly design the cache strategies for each BS.

3 Problem Analysis and Formulation

As stated in system model, we aim to design a global cache strategy to minimize the overall transmission delay.

The wireless link delay for content transmission between BS m and user u can be defined as:

$$d^{dir}_{u,m} = \frac{Z}{R_{mu}} = \frac{Z}{B\log(1+\frac{P_t l^{-\alpha}_{um}}{\sigma^2})}, \tag{4}$$

Considering the user u cache, when the user requests content f in time slot t, the wireless link delay is:

$$d^{dir}_{m,u,f} = \begin{cases} d^{dir}_{u,m} & g^t_{u,f} = 0 \\ \dfrac{K-1}{K} d^{dir}_{u,m} & g^t_{u,f} = 1. \end{cases} \tag{5}$$

Meanwhile, the set of wireless link delay between BSs and users can be defined as: $\zeta = \{\frac{Z}{R_{mu}} | \forall u \in \Phi^{(m)}_U, \forall m \in \Phi_M\}$. The cooperation delay of user u through the cooperation request content f between BSs can also be similarly expressed as:

$$d_{u,f}^{co} = \begin{cases} \dfrac{Z}{R_{co}} & g_{u,f}^{t} = 0 \\ \dfrac{K-1}{K}\dfrac{Z}{R_{co}} & g_{u,f}^{t} = 1. \end{cases} \tag{6}$$

Assuming that there are two processes to fetch a content from the cloud server: fetch and download delay is correspondingly defined with d^{fetch} and d^{down}, respectively. In particularly, the download time will be much larger than the wireless latency *i.e.,* $d^{down} \gg \frac{Z}{R_{mu}}$. d_{fetch} will be inevitable unless the user caches the entire content (*i.e.,* $K = 1$). Therefore, the backhaul delay for user m to download content f from the cloud server through the BS is:

$$d_{u,f}^{backhaul} = \begin{cases} d^{fetch} + d^{down} & g_{u,f}^{t} = 0, K \geq 1 \\ d^{fetch} + \dfrac{K-1}{K} d^{down} & g_{u,f}^{t} = 1, K > 1 \\ 0 & g_{u,f}^{t} = 1, K = 1 \end{cases} \tag{7}$$

Based on (Eqs. 5, 6, 7) and caching model, in time slot t, the transmission delay for user u to request a content f through its serving BS m is:

$$d_{m,u,f}^{t} = \begin{cases} d_{m,u,f}^{dir} & a_{m,f}^{t} = 1 \\ d_{m,u,f}^{dir} + d_{u,f}^{co} & \exists a_{m_{co},f}^{t} = 1, m_{co} \in \Phi_M \\ d_{m,u,f}^{dir} + d_{u,f}^{backhaul} & others. \end{cases} \tag{8}$$

Therefore, the overall transmission delay in time slot t can be expressed by:

$$\mathbb{D}^{t}(\mathbb{A}^{t}, \mathbb{B}^{t}, \mathbb{G}^{t}) = \sum_{m}^{M} \sum_{u}^{U_m} \sum_{f}^{F} d_{m,u,f}^{t} b_{u,f}^{t}. \tag{9}$$

In order to minimize the long-term network transmission delay $\mathbb{D}^{t}(\mathbb{A}^{t}, \mathbb{B}^{t}, \mathbb{G}^{t})$, the hybrid-cache centralize problem can be formulated as:

$$\begin{aligned} \mathbf{P1}: \min_{\mathbb{A}^t} \quad & \sum_{t=1}^{T} \mathbb{D}^{t}(\mathbb{A}^{t}, \mathbb{B}^{t}, \mathbb{G}^{t}) \\ \text{s.t.} \quad & \textstyle\sum_{f=1}^{F} a_{m,f}^{t} \leq C_{BS}/Z, t = 1, 2, \cdots, T \forall m \in \Phi_M. \\ & a_{m,f}^{t} \in \{0, 1\}, t = 1, 2, \cdots, T \quad \forall f \in \mathcal{F}, \forall s \in \Phi_S. \end{aligned} \tag{10}$$

Obviously, $\mathbb{A}^t$ in **P1** is a high-dimensional discrete variable, and the delay in $D^{t}(\mathbb{A}^{t}, \mathbb{B}^{t}, \mathbb{G}^{t})$ is continuous. Therefore, **P1** is a NP-hard problem and complexity of solving non-convex **P1** with traditional optimization algorithms will be extremely high. To achieve the design goal in (10), the CU should improve the caching hit-ratio (*i.e.,* the ratio of the number of request contents that are cached over the total number of request contents) for each BS and reduce delays by designing coordination strategy among BSs.

4 Proposed Method and Algorithm

Reinforcement learning has a strong ability to adapt to the environment, but when the input data has a longtail distribution, the neural network responsible for the policy will still rely the most popular contents.

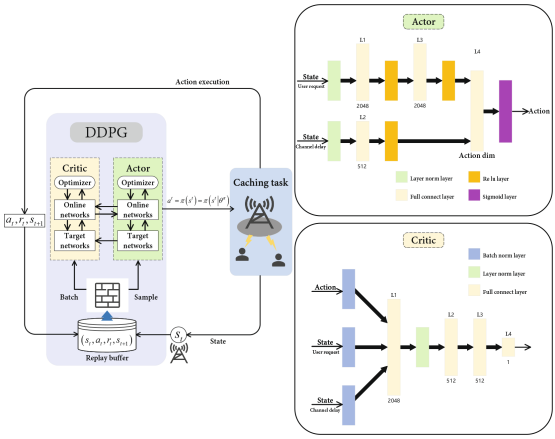


Fig. 2. Proposed DDPG-based Anti-longtail Algorithm.

In this work, we use DDPG as the baseline model, and by referring to deep long-tail learning, we propose a reinforcement learning algorithm dedicated to wireless caching. As illustrated in Fig. 2, is the algorithm architecture of DDPG. DDPG consists of two parts, actor and critic, both of which are composed of deep neural networks.

4.1 Reinforcement Learning

Reinforcement learning framework is generally defined based on Markov decision process(MDP). MDP is usually including three base components: (S, A, R), which is define as state space, action space and rewards respectively. Correspondingly, the system state and action in time slot t can be defined as $s^t \in \mathcal{S}$ and $a^t \in \mathcal{A}$. To be precise, the agent will observe the environment state s^t in each time slot t and make an action with policy function $\pi^t(a|s)$ to maximum the reward r^t.

The state, action and reward are accordingly formulated in our proposed DRL and introduced as follows: **System state**: According to optimization problem (**P1**), in order to further reduce the transmission delay in this work, the BS cache strategy needs to be designed according to user requests and the channel conditions of different users. Since users request the same set of contents, $\mathbb{B}^t$ can be reduced to $\mathbb{H}^t = \left\{\sum_u^U b_{u,f}^t\right\}^{1\times F}$: The system state is defined as user request and current BS - User channel state: $s^t = (\mathbb{H}^t, \mathbb{G}^t)$. **Action space**: In propose

DRL, the action is defined as the optimization variable in (**P1**). So the system action: $a^t = \mathbb{A}^t$. **Reward**: We normalize the total delay value in a time slot t as reward: $r(s^t, a^t) = e^{-\mathbb{D}^t(\mathbb{A}^t, \mathbb{B}^t, \mathbb{G}^t)}$. Latency caused by the most popular contents is further reduced due to the presence of user caches. Less popular contents will be an important factor influencing further declines in latency. Therefore, some of the most popular contents and the less popular contents will become the primary goals of agent optimization.

To further optimize (**P1**), we will introduce a Q-value function to optimize the long-term delay with T time slot in one epoch:

$$Q(s^t, a^t) = \mathbb{E}_\pi \left[\sum_{t'}^{T} r(s^{t'}, a^{t'}) | s^t, a^t \right] \tag{11}$$

So the policy function will take the action that maximizes the Q-value function:

$$\pi(s^t) = \arg\max_{a^t} Q(s^t, a^t). \tag{12}$$

4.2 Proposed DDPG-Based Anti-longtail Algorithm

The architecture of DDPG is shown in Fig. 2(a). There are five components in DDPG: actor network, actor target network, critic network, critic target network and replay buffer. The replay buffer will record each step of the reinforcement learning algorithm in the form of (s^t, a^t, r^t, s^{t+1}).

The action selection is given by the actor network. Assuming that the parameter of the actor network is θ^π, then: $a^t = \pi(s^t) = \pi(s^t|\theta^\pi)$, the actor target network extracts the state s' from the replay buffer and gets the action a', assuming its parameter is $\theta^{\pi'}$ then $a' = \pi'(s') = \pi(s'|\theta^{\pi'})$.

The role of the critic is to evaluate the actor network, and the critic is to fit and implement the Q-value function. Assuming that the parameter of the critic network is θ^Q, then the Q-value function is: $Q(s^t, a^t) = Q(s^t, a^t|\theta^Q)$. The critic target network evaluates the target Q-value Q' from the next state extracted from the replay buffer and the output of the actor target network. Assuming that its parameter is $\theta^{Q'}$, there is $Q'(s', a')) = Q'(s', a'|\theta^{Q'})$. So the loss function of critic network is the mean squared error(MSE) between Q' output by the critic target network plus reward and the output of the critic network:

$$\mathcal{L}_{critic} = \frac{1}{N} \sum_t (r^t + \delta Q'(s^{t+1}, \pi'(s^{t+1}|\theta^{\pi'})|\theta^{Q'}) - Q(s^t, a^t|\theta^Q))^2, \tag{13}$$

where N is the size of the mini-batch extracted from the replay buffer, and δ is the discount factor, whose function is to make the Q-value pay more attention to the current environmental state. Therefore, the critic network can be updated as:

$$\theta^Q = \theta^Q - \beta_{critic}\mathcal{L}_{critic}, \tag{14}$$

where β_{critic} is the learning rate of critic. The loss function of actor network will be given by critic network:

$$\mathcal{L}_{actor} = \frac{1}{N}\sum_t \nabla_a Q(s^t, a^t|\theta^Q) \times \nabla_{\theta^\pi}\pi(s^t|\theta^\pi). \tag{15}$$

The network parameters of the actor network can be updated as:

$$\theta^\pi = \theta^\pi - \beta_{actor}\mathcal{L}_{actor}. \tag{16}$$

The parameter of actor target network and critic target network are sync with actor and critic, respectively. Assuming that the soft update factor is τ, the update process can be express as:

$$\begin{aligned} \theta^{\pi'} &= \tau\theta^\pi + (1-\tau)\theta^{\pi'} \\ \theta^{Q'} &= \tau\theta^Q + (1-\tau)\theta^{Q'}. \end{aligned} \tag{17}$$

Actor network architecture is shown in Fig. 2(b). Since the state contains discrete user requests and continuous wireless channels, it is isolated at the entrance of the network, and then a layer of layer normalize is passed to normalize the parameters. At the output end of the network, the network output is discrete by the sigmoid function, and the caching strategy of each BS is obtained.

The critical network architecture is shown in Fig. 2(c). Since the state contains discrete user requests, the BS caching strategy is associated with continuous wireless channels. Therefore, batch normalize is also done at the entrance, and after passing through independent fully connected layers, they are concentrate together. What's more, in order to prevent over-fitting when there are too many most popular contents, we add L2 regularization to make a new loss function $\mathcal{L}'_{critic}$:

$$\mathcal{L}'_{critic} = \mathcal{L}_{critic} + \lambda||\theta^Q||^2, \tag{18}$$

where λ is the penalty factor. So far, the algorithm flow of our proposal is:

Algorithm 1. Proposed DDPG-base anti-longtail algorithm

Require:
β_{actor}, β_{critic}: learning rate for actor and critic; BS set Φ_S, User set Φ_U, $\Phi_U^{(s)}$;
randomly Initialize θ^{π} and θ^{Q}
Initialize target network $\theta^{\pi'} = \theta^{\pi}$ and $\theta^{Q'} = \theta^{Q}$
Create an empty replay buffer $\mathbb{D}$
Initialize a random system state $s^0 = (\mathbb{H}^0, \mathbb{G}^0)$
for each epoch **do**
  for t=0,T **do**
    Select action $a^t = \pi(s^t|\theta^{\pi})$ and transfer to caching strategy $\mathbb{A}^t$ with limited capacity of BS.
    According to the caching strategy $\mathbb{A}^t$, and system state $s^t = (\mathbb{H}^t, \mathbb{G}^t)$, calculate the overall delay using (9) to observe the reward $r^t = r(s^t, a^t) = e^{-\mathbb{D}^t(\mathbb{A}^t, \mathbb{B}^t, \mathbb{G}^t)}$.
    Observe new state s^{t+1}.
    Store trajectory (s^t, a^t, r^t, s^{t+1}) in $\mathbb{D}$
    $\theta^{\pi'}$, θ^{π}, $\theta^{Q'}$, θ^{Q} = DDPG-Learning($\mathbb{D}$,$\theta^{\pi'}$,θ^{π},$\theta^{Q'}$,θ^{Q})
  end for
end for

procedure DDPG-Learning($\mathbb{D}$, $\theta^{\pi'}$, θ^{π}, $\theta^{Q'}$, θ^{Q})
  Sample a random mini-batch of N trajectories from $\mathbb{D}$
  Minimize the critic loss $\mathcal{L}_{critic}$ using (13) or (18)
  Update critic network θ^{Q} using (14)
  Minimize the critic loss $\mathcal{L}_{actor}$ using (15)
  Update critic network θ^{π} using (16)
  Soft update the target network using (17):
  $\theta^{\pi'} = \tau\theta^{\pi} + (1-\tau)\theta^{\pi'}$
  $\theta^{Q'} = \tau\theta^{Q} + (1-\tau)\theta^{Q'}$
  return $\theta^{\pi'}$, θ^{π}, $\theta^{Q'}$, θ^{Q}
end procedure

5 Performance Evaluation

In this part, the numerical simulations are performed to demonstrate the performance of propose method. In order to facilitate discussion purposes, the cache hit rate is divided into two parts: direct hits indicate that user requests can be served by the serving BS and coordinated hits indicate that user requests can be served by coordinated BSs with extra delay. In this way, the user's cache does not directly affect the cache hit rate.

The bandwidth is set as 40 MHz, while the path-loss exponent α is set as 2.5, transmit power and noise power are set as 40 dBm and -120 dBm/Hz, respectively. To verify its effectiveness, we use the commonly adopted deep Q-learning (DQN) algorithm as a comparison benchmark while ensuring that the data is consistent in the comparison. The In DQN method, the neural network in DQN use MLP and Adam optimizer. The structure and parameter of DQN neural network keep the same with propose actor network, without normalize layer (Batch-normalize Layer and Layer-normalize Layer). In order to verify the promotion effect of the hybrid-caching model on method, there are a group DQN will use the normal cache model (without user partial cache).

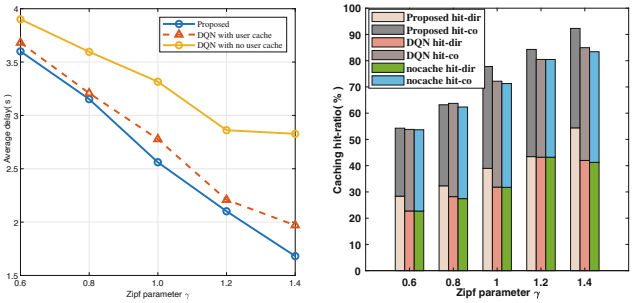
Assume that distance between adjacent BS is set as 500 m and the BS serving distance l_p is set as 250 m. The remaining parameters are given in Table 1.

Table 1. Simulation settings.

Attribute	Value
Number of BSs	4
Number of users	120
F (number of content set)	400
C (Capacity of each BS)	50
γ (Zipf parameter)	1
Batch size	2048
β_{actor} (Actor learning rate)	0.01
β_{critic} (Critic learning rate)	0.02
Z (size of content)	5×10^8 bits
d_0 (backhaul delay)	4 s
d_{fetch} (fetch contents delay)	0.4 s
δ (discount factor)	0.96
τ (soft update parameter)	0.02
C_{real} (user physical capacity)	5
K	2
λ	10^{-4}

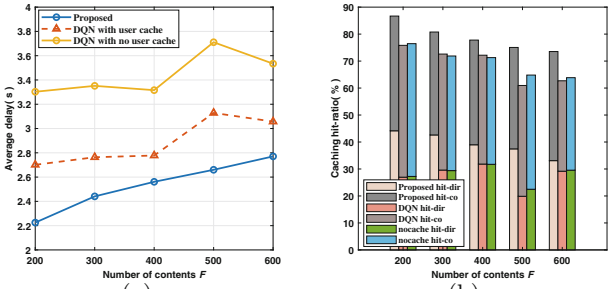
First, we consider the delay and caching hit rate comparison between proposed method and compared algorithms for different Zipf parameter γ, where $F = 400$ is set. As seen in Fig. 3, the propose method consistently maintains the lowest latency. Specially, as the γ increases, the delay gap increases, which show the influence of Normalize Layer and L2 regularization in proposed method on long-tail distribution. In the cache hit rate, DQN has a weak advantage over the no user cache case, which indicates that user cache does not affect the cache hit rate in method comparison. This proves the algorithmic advantage of the propose method on DRL.

Then, we consider the impact of the content-set size on the compared algorithms. As can be seen in Fig. 4, as the content set becomes larger, the request probability of popular contents leaks slowly (from 200 to 1000, the request probability of the top 100 contents decreases by 18.9%). So the less popular contents allocation will greatly affect the latency performance. In addition, the unpopular contents will interfere with the convergence of the algorithm. The propose one, the delay is slowly increasing as the contents set gets larger. In contrast, DQN appears unstable and falls into a local optimum, especially when $F = 500$. In fact, when $F = 200$, through the cooperation between BSs, the algorithm has the theoretical possibility of caching the entire content set. It can be seen that only the propose method has a high overall cache hit ratio.



(a) (b)

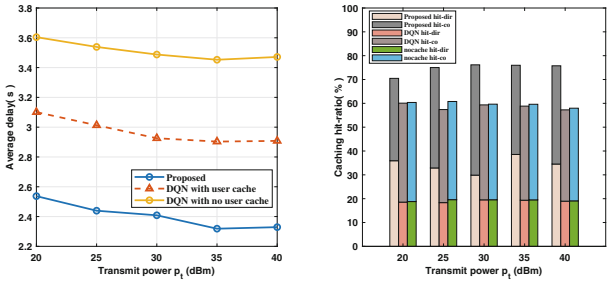
Fig. 3. Delay and caching hit rate comparison with different γ where $l_{\max} = 500\,\mathrm{m}$.



(a) (b)

Fig. 4. Delay and caching hit rate comparison for different F.

In Fig. 5, we present the comparison of average delays between the proposed and DDPG with different transmit power, where $F = 400$ and $\gamma = 1$. It can be seen that, each of the three algorithms has a slight delay drop and is relatively stable. On the one hand, it shows the strong adaptability of reinforcement learning to the environment and the generalization ability of the algorithm. On the other hand, it shows that in the wireless cache, backhaul traffic is still the major cause of delay. In fact, the overall delay of the wireless link is also significantly reduced (the number of users is 120).



(a) (b)

Fig. 5. Delay and caching hit rate comparison for different p_t.

6 Conclusion

We studied the edge caching design under the multi-BS cooperate hybrid-cache model. A DDPG-based Anti-longtail method is proposed to cope with the impact on DRL with longtail input. Particularly, the long-tail request distribution will make the neural network depend on the most popular content after gradient descent, which will affect the deployment and training of the less popular content. Through the transformation of the algorithm and the proposed hybrid cache model, we further reduce the transmission delay in the wireless cache. Numerical simulations show that this method outperforms commonly used DRL methods in terms of cache hit rate and overall transmission delay.

Acknowledgement. The work was supported in part by National Key Research and Development Program of China (No. 2021YFB2012403), in part by the Science and Technology Program of Guangzhou (No. 202201020182) and in part by the Guangdong Basic and Applied Basic Research Foundation (No. 2021A1515011812).

References

1. Barnett, T., Jain, S., Andra, U., Khurana, T.: Cisco visual networking index (VNI) complete forecast update, 2017–2022. Americas/EMEAR Cisco Knowledge Network (CKN) Presentation, pp. 1–30 (2018)
2. Sheraz, M., et al.: Artificial intelligence for wireless caching: schemes, performance, and challenges. IEEE Commun. Surv. Tutorials **23**(1), 631–661 (2021)
3. Zhang, Z., Tao, M.: Deep learning for wireless coded caching with unknown and time-variant content popularity. IEEE Trans. Wireless Commun. **20**(2), 1152–1163 (2021)
4. Chaudhary, A., Gupta, H.P., Shukla, K.K.: Real-time activities of daily living recognition under long-tailed class distribution. IEEE Trans. Emerging Top. Comput. Intell. **6**(4), 740–750 (2022)
5. Zhang, Y., Kang, B., Hooi, B., Yan, S., Feng, J.: Deep long-tailed learning: a survey. arXiv preprint arXiv:2110.04596 (2021)

6. Chen, D., Liu, Y.-C., Kim, B., Xie, J., Hong, C.S., Han, Z.: Edge computing resources reservation in vehicular networks: a meta-learning approach. IEEE Trans. Veh. Technol. **69**(5), 5634–5646 (2020)
7. Alshammari, S., Wang, Y.-X., Ramanan, D., Kong, S.: Long-tailed recognition via weight balancing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6897–6907 (2022)
8. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning. In: ICLR (Poster) (2016)