Cooperative Cache in Cognitive Radio Networks: A Heterogeneous Multi-Agent Learning Approach

Ang Gao, Hengtong Liu, Yansu Hu, Wei Liang, Soon Xin Ng

***Abstract*—Deploying distributed cache in cognitive radio networks (CRNs), which spreads popular contents to the edge of network during the off-peak time through spectrum sharing, can reduce the deliver delay to users nearby without causing severe interference to the primary network. However, due to the un-predicable contents requirement as well as the band occupation of primary users, it is non-trivial to optimize the cache storage and contents fetching strategy of users dynamically. The paper proposes a heterogeneous multi-agent deep deterministic policy gradient (MADDPG) approach, which takes users and cache servers as two different types of agents to learn the cooperation and competition for mutual benefits. The numeral simulation demonstrates that comparing with the other single or homogeneous deep reinforcement learning (DRL) approaches, the proposed heterogeneous MADDPG can further reduce the delivery delay of users and enhance the cache efficiency of SBSs.**

***Index Terms*—Cognitive Radio Networks, Cooperative Cache, Multi-Agent Deep Deterministic Policy Gradient**

I. INTRODUCTION

COGNITIVE radio networks (CRNs) provide a constructive prospect for improving spectrum efficiency and alleviating spectrum scarcity by allowing unlicensed secondary network to opportunistically access the spectrum licensed to primary network [1]. Cooperative cache, equipped at the secondary base-stations (SBSs) in CRNs [2] can spread contents to the edge of primary network in exchange for additional spectrum accessing opportunities. In specific, SBSs acting as edge cache servers proactively fetch hot contents from the primary base-station (PBS) during the off-peak time and serve the users nearby through the secondary network, which can alleviate the interference, decrease the traffic loads, and reduce the contents delivery delay [2]. As a result, mutual benefits can be gained by both the primary and secondary network.

The effective exploitation for edge cache highly depends on the contents popularity. However, it changes dynamically in both temporal and spatial which is unknown in advance [3]. To this end, machine learning has been adopted to predict the contents popularity based on the historical observation and optimize the cache storage. For example, paper [4] models the cache updating problem as a Markov decision process (MDP) and takes the long short-term memory (LSTM) based deep Q-learning network (DQN) to optimize the cache replacement strategy in multiple small cells networks. It is a off-policy based centralized learning approach that needs redundant communication overhead. Paper [1] takes the multi-user multi-cache contents delivery networks (CDNs) as a single global agent, and optimizes both the cache storage and users' contents fetching strategy to minimize the system cost by LSTM based deep deterministic policy gradient (DDPG). Paper [5] also adopts DDPG based approach to maximize the long-term hit rate and minimize their front haul traffic loads. Paper [6] further employs the homology optimization to reform the reward function to guarantee that the network parameters can be efficiently updated during the training. Papers [5], [6] take each cache node as an individual agent with a global critic network to evaluate the cache action. However, they lose sight that both users and cache servers are all autonomous agents, which means users tends to fetch the content from the cache node with the minimum delivery delay, while cache nodes expect to storage popular contents to attract more users for the maximum reputation or revenue [7].

The paper focuses on the following three issues by adapting heterogeneous multi-agent DDPG (MADDPG) approach in multi-cache CRNs. ① For cache nodes, how many percentages of a given content should be cached to maximize the storage efficiency? ② For users, which cache node is the best choice for contents fetching to minimize the delivery delay? ③ How to design an effective learning algorithm for heterogeneous agents with different action and observation space in a cooperation and competition co-existing environment?

The main contributions of the paper are as follows:

- Different from the previous researches which take multi-cache as homogeneous agents [5], [6], or model the whole system as a single global agent [4], the paper proposes a heterogeneous MADDPG approach, where both users and SBSs are treated as two types of agents interacting with each other, to reduce the average delivery delay of users and enhance the cache efficiency of SBSs at the same time.
- Unlike DQN[4] that only operates in a discrete action space, by the proposed MADDPG, SBSs can optimize the cached fraction of each content in an independent and consecutive space rather than simply evicting or retaining the whole content, which makes a flexible usage of cache and enhances the storage efficiency.
- With the centralized-training and decentralized-execution features of MADDPG, the well-trained agents can optimize their action individually by partial observation, i.e. the user fetches contents from SBSs leading to the minimum delay, while SBSs optimize the storage strategy to maximum the cache efficiency, which can reduce the synchronization and communication overhead.

The work was supported by Shaanxi Natural Science Foundation 2021GXLH-01-15, 2021JM-186. Xian Technology Plan 21RGZN0018 and Taicang Keypoint Science and Technology Plan TC2019SF03. (*Corresponding author: Ang Gao, e-mail: gaoang@nwpu.edu.cn*)

Ang Gao, Hengtong Liu and Wei Liang are with the School of Electronics and Information, Northwestern Polytechnical University, 710072, China.

Yansu Hu is with the School of Electronics and Control Engineering, Chang'an University, 710064, China.

Soon Xin Ng is with the School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K.

II. Problem Formulation

A. System Model

As shown in Fig.1, there are one PBS that contains all contents, and $\mathcal{N} = \{1, \cdots, N\}$ SBSs as edge cache servers with the caching size C_n ($n \in \mathcal{N}$), and $\mathcal{K} = \{1, \cdots, K\}$ users[1]. Each user/SBS is equipped with a single omni-directional antenna. In general, users first fetch a content (or part of it) from an appropriate SBS by the secondary network. If the content (or rest of it) is not stored in SBSs, users further access PBS by the primary network. By the SBS-first policy, the spectrum resource of primary network can be efficiently freed up and the average delay of the secondary network is decreased [3]. Thus, cooperatively caching popular contents in SBSs during the off-peak time of the primary network can make mutual benefit for both the primary and secondary network.

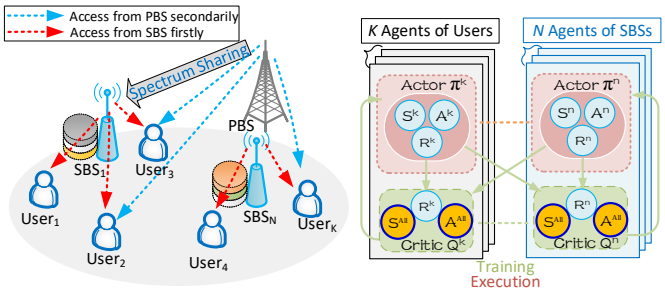

Fig. 1: System model

Assuming the overall contents consist of F files, and the file size s_f, $f \in \mathcal{F}$ is independent identically distributed (i.i.d) of Pareto distribution with the average value of $\bar{s}$, where $\mathcal{F} = \{1, \cdots, F\}$. Let $\boldsymbol{q}_k(t) = [q_{k,1}(t), \cdots, q_{k,F}(t)]$ be the contents request vector with Zipf distribution [5], where $q_{k,f}(t) \geq 0$ is the number of times that user k requests file f at time slot t. Although the contents are modeled to be static in this paper, the number of requests $q_{k,f}$ is time variant with respect to users.

Each file can be divided into fragments by using a rateless code like Raptor codes [8]. Let $\Lambda_n(t) = [\lambda_{n,1}(t), \cdots, \lambda_{n,F}(t)]$, and $\lambda_{n,f}(t) \in [0, 1]$ denote the fraction size of file f cached at SBS n at time slot t and $\lambda_{0,f} \equiv 1$, i.e., PBS contains all fragments of files. Due to the fact that the coded fragments are usually randomly spread to different SBSs, the contents overlap among different SBSs may restrain the cooperative cache from being effectiveness [9]. With this concern, for user k, it can only retrieve one fragment of file f from SBS n with the size of $\lambda_{n,f}$, and retrieve the rest fraction with the size of $1-\lambda_{n,f}$ from PBS relying on primary transmission, rather than employing multiple SBSs to cooperatively relay the fragments [10]. Because SBSs are geographically closer to users than PBS.

Let $d_{n,k}$ be the distance between user k and SBS n and the channel gain is $h_{n,k} = \rho_0 d_{n,k}^{-\alpha}$. Thus the transmission rate is $r_{n,k} = B \log_2(1 + \frac{p_n h_{n,k}}{\delta^2})$, where p_n is the transmission power of SBS n and δ^2 is the power of addictive white Gaussion noise (AWGN). Since SBSs are geographically closer to users compared to PBS, the transmission rate of SBSs is faster than that of PBS, i.e. $r_{n,k} > r_{0,k}, \forall n \in \mathcal{N}, \forall k \in \mathcal{K}$. The slot duration T_0 is long enough[2] that all contents request of users can be served within a time slot.

[1]Both the primary and secondary users can fetch contents from SBSs nearby for a better transmission rate and less delivery delay. Thus we do not distinguish the primary and secondary users in the model any more.

B. Delay Optimization of Users

Let $\mathbf{a}_k(t) = [a_{k,1}(t), \cdots, a_{k,F}(t)]$ be the fetching vector, which indicates that user k will fetch file f with the fraction of $\lambda_{a_{k,f},f}$, from server $a_{k,f} \in \mathcal{N}^+$ at time slot t, where $\mathcal{N}^+ = \{\mathcal{N} \cup 0\}$ represents the joint set of cache servers (including PBS and SBSs), and $a_{k,f} = 0$ indicates users k fetches content f from PBS. Each user pursuits to find the best set of $\mathbf{a}_k(t)$ to minimize the overall delivery delay, i.e. the delay summation of all F files with respect to $\Lambda_n(t)$.

$$\ell_k(\mathbf{a}_k, \Lambda_n) = \sum_{f=1}^{F} \gamma_{k,f}, \tag{1}$$

$$\gamma_{k,f}(a_{k,f}, \lambda_{n,f}) = [\underbrace{\frac{\lambda_{a_{k,f}(t),f}(t)}{r_{a_{k,f},k}}}_{①} + \underbrace{\frac{1 - \lambda_{a_{k,f}(t),f}(t)}{r_{0,k}}}_{②}] s_f q_{k,f}(t). \tag{2}$$

Eq.(2) is composed of two items, where ① is the delay to fetch file f with the fraction of $\lambda_{a_{k,f},f}$ from SBS $a_{k,f}$, and ② denotes the delay to fetch the rest fraction $(1 - \lambda_{a_{k,f},f})$ from PBS. Thus the optimization problem for users with the consideration of fairness is to minimize the maximum expected delivery delay:

$$\text{P1}: \min_{\mathbf{a}_1, \cdots, \mathbf{a}_K} \max \Big[\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \ell_k(\mathbf{a}_k, \Lambda_n) \Big], \tag{3}$$

$$\text{s.t.} \quad a_{k,f}(t) \in \{0, N\}. \tag{C1}$$

C. Cache Efficiency Optimization of SBSs

1) Contents Popularity: Let $\mathbf{p}_n(t) = [p_{n,1}(t), \cdots, p_{n,F}(t)]$ be the popularity vector, where $p_{n,f}(t)$ is the popularity of file f observed by SBS n at time slot t, i.e. the number of requested times of file f at the latest time slot.

$$p_{n,f}(a_{k,f}) = \frac{\sum_{k=1}^{K} \mathcal{I}_n(a_{k,f}(t)) q_{k,f}(t)}{T_0}, \tag{4}$$

where $\mathcal{I}_n$ is the indication function that implies whether file f is fetched from SBS n or not, i.e.,

$$\mathcal{I}_n(a_{k,f}(t)) = \begin{cases} 1, & a_{k,f}(t) = n, \\ 0, & otherwise. \end{cases} \tag{5}$$

Thus the files popularity observed by SBS n is the function of users' fetching action $a_{k,f}$, i.e., Eq.(4) is the number of requested times of file f at SBS n in the time duration T_0 .

2) Incentive Mechanism: The incentive mechanism is an effective way to encourage SBSs taking fully usage of the storage [11], i.e., SBSs optimize their storage strategy to attract more fetching for better revenue with a given incentive price. SBSs will get a positive revenue when files are successfully fetched by users (also known as cache hit). The revenue is directly proportional to the cumulative size of fetched files.

$$u_n^{hit}(a_{k,f}, \lambda_{n,f}) = \epsilon \sum_{f \in \mathcal{F}} \underbrace{\Big(\sum_{k \in \mathcal{K}} \mathcal{I}_n(a_{k,f}(t)) q_{k,f}(t) \Big)}_{\text{fetching times for the file } f} \overbrace{s_f \lambda_{n,f}(t)}^{\text{file size}}, \tag{6}$$

where ϵ is the unit incentive price measured by \$/Mbits.

3) Cache Replacement: The cache storage limited by C_n should be refreshed from time to time. To maximize the cache utilization, SBSs tend to store popular contents as many as possibly. The replacement cost arises when SBSs request contents from PBS to increase the files storage fraction, because it may lead to extra delay and energy cost. There is no replacement cost when SBSs reduce the caching fraction. So the cache replacement cost at time slot t is defined as:

$$c_n^{replace}(\lambda_{n,f}) = \sum_{f\in\mathcal{F}} s_f \max\{\lambda_{n,f}(t) - \lambda_{n,f}(t-1), 0\}. \quad (7)$$

The reward of SBS n at time slot t can be calculated as:

$$u_n(a_{k,f}, \lambda_{n,f}) = u_n^{hit}(a_{k,f}, \lambda_{n,f}) - \beta c_n^{replace}(\lambda_{n,f}), \quad (8)$$

where β is the weight factor of replacement cost with unit of \$/Mbits. To take fully usage of cache storage, the contents should be spread among SBSs considering the max-min fairness. So the optimization problem for multiple SBSs is:

$$\text{P2}: \max_{\Lambda_1,\cdots,\Lambda_N} \quad \min\Big[\lim_{T\to\infty}\frac{1}{T}\sum\nolimits_{t=1}^{T} u_n(\mathbf{a}_k, \Lambda_n)\Big], \quad (9)$$

$$\text{s.t.} \quad \sum\nolimits_{f=1}^{F} \lambda_{n,f}(t) \le C_n. \quad (C2)$$

$$\lambda_{n,f}(t) \in [0,1]. \quad (C3)$$

III. Heterogeneous Multi-Agent Deep Reinforcement Learning

A. Cooperation and Competition among Users and SBSs

P1 for users and P2 for SBSs are mutual related. In specific, $\ell_k(\mathbf{a}_k, \Lambda_n)$ in P1 is the function of $\lambda_{n,f}$ which is the optimized variables of P2. Similarly, $u_n(\mathbf{a}_k, \Lambda_n)$ in P2 is the function of $a_{k,f}$ which is just the optimized variables of P1. Besides, P1 and P2 are mixed integer non-linear problems (MINPs) due to $a_{k,f}$ and $\sum_{k\in\mathcal{K}} \mathcal{I}_n(a_{k,f})\lambda_{n,f}$ in Eq.(6). Hence they are hard to be solved by successive convex approximation (SCA) method. Meanwhile, there exist both competition among homogeneous agents and cooperation among heterogeneous agents.

- Now that users tend to fetch a file from SBSs with the maximum $\lambda_{n,f}/r_{k,n}$ according to Eq.(2) to minimize its delivery delay, SBSs should cache the 'hottest' contents as many as possible and *compete* to attract more users to enlarge the revenue according to Eq.(6).
- The storage strategy $\lambda_{n,f}$ of SBSs and the fetching choice $\mathbf{a}_k$ of users react upon each other, which means they should *cooperate* in mutual to enhance the reward.

B. Multi-Agent Learning Model

For such a mixed cooperative-competitive environment [12], each agent is selfish and rational, they only pursuit to maximize their own reward. The paper proposes a multi-agent learning approach, where agents can autonomously seek to balance the delivery delay of users (P1) and the cache efficiency of SBSs (P2) all by themselves. As shown in Fig.1, there are two types of agents, i.e., user-type and SBS-type. Although the diagrams of them are similar, their action, observation as well as the reward are constructed with different meaning detailed as follows:

- $\mathcal{S}_t^k = \{\mathbf{q}_k(t)\}$ denotes the requested times of each file at time slot t observed by user k and $\mathcal{S}_t^k \in \mathcal{Z}^F$, where $\mathcal{Z}$ is the sets of natural number.
- $\mathcal{A}_t^k = \{\mathbf{a}_k\}$ denotes the caching index, i.e. user k fetches the first fraction of file f from cache node $a_{k,f} \in \mathcal{N}^+$ (including SBSs and PBS) and $\mathcal{A}_t^k \in \mathcal{N}^{+F}$.
- $\mathcal{R}_t^k = -\ell_k(t)$ is the scaler reward of users k, i.e. the negative value of delay defined by Eq.(1).
- $\mathcal{S}_t^n = \{\mathbf{p}_n(t)\}$ denotes the contents popularity observed by SBS n, i.e. $p_{n,f} \in [0,1]$ and $\mathcal{S}_t^n \in [0,1]^F$.
- $\mathcal{A}_t^n = \{\Lambda_n(t)\}$ is the fraction vector of contents cached in SBS n, i.e. $\lambda_{n,f} \in [0,1]$ and $\mathcal{A}_t^n \in [0,1]^F$.
- $\mathcal{R}_t^n = u_n(t)$ is the scalar reward of SBS n defined in Eq.(8).

Heterogeneous MADDPG is enforced by centralized training (red and green data flow) and decentralized execution (green data flow) in Fig.1. Each agent has an actor and a critic to ensure the learning stability and overcome the overoptimistic in large-scale problem [13]. During the training, the critic collects the global observation $\mathcal{S}^{\text{All}} = [\mathcal{S}_{n\in\mathcal{N}}^n : \mathcal{S}_{k\in\mathcal{K}}^k]$ and the global action $\mathcal{A}^{\text{All}} = [\mathcal{A}_{n\in\mathcal{N}}^n : \mathcal{A}_{k\in\mathcal{K}}^k]$ to evaluate the fitness $\mathcal{A}^n$ of SBSs or $\mathcal{A}^k$ of users, and generate the policy gradient to guide the evolution of actor network [2]. During the execution, the well-trained actor makes the optimal action ($\mathcal{A}^n$ or $\mathcal{A}^k$) independently without referring to other agents' action or state, which leads to a less communication overhead.

MADDPG further adds noise $\mathbb{N}$ to explore better strategies. Another essential technique is the experience replay buffer (RB). Each agent is equipped with a RB to store the (state, action, reward, next state) transition $(\mathcal{S}_t^{\text{All}}, \mathcal{A}_t^{\text{All}}, \mathcal{R}_t^k, \mathcal{S}_{t+1}^{\text{All}})$, which will be fetched out randomly to update the parameters. Together with mini-batch, experience replay can effectively avoid the highly correlated action for successive updating. For clarity, the superscript $m \in \{\mathcal{N}\cup\mathcal{K}\}$ is used to denote the number of agents without distinguishing SBSs or users.

- Actor online $\pi^m(\mathcal{S}^m|\mu^m)$: generates action $\mathcal{A}^m = \pi^m(\mathcal{S}^m|\mu^m)$ and updates parameter μ^m in gradient direction, where $\mu_{t+1}^m = \mu_t^m + \alpha_\mu \nabla_\mu J(\mu^m)$ and α_μ is the updating step size.
- Actor target $\pi'^m(\mathcal{S}^m|\mu'^m)$: generates the optimal next action $\mathcal{A}'^m_{t+1} = \pi'^m(\mathcal{S}^m_{t+1}|\mu'^m)$ according to the next state $\mathcal{S}^m_{t+1}$ from either the environment or the experience RB.
- Critic online $Q^m(\mathcal{S}^{\text{All}}, \mathcal{A}^{\text{All}}|\theta^m)$: calculates θ^m to minimize the temporal difference error L according to the Q^m value of each action-state pair $(\mathcal{S}^{\text{All}}, \mathcal{A}^{\text{All}})$ from the mini-batch.

$$L = \frac{1}{\mathbb{M}}[\sum(y_t - Q^m(\mathcal{S}_t^{All}, \mathcal{A}_t^{\text{All}}|\theta^m))^2], \quad (10)$$

$$\theta_{t+1}^m = \theta_t^m + \alpha_{\theta^Q} \nabla_{\theta^Q} Q^m(\mathcal{S}_t^{\text{All}}, \mathcal{A}_t^{\text{All}}|\theta^m), \quad (11)$$

where α_θ is the step size for parameters updating, and y_t^m is calculated by the critic target network as:

$$y_t^m = \mathcal{R}_t^m + \gamma Q'^m\left(\mathcal{S}_{t+1}^{\text{All}}, \mathcal{A}'^{\text{All}}_{t+1}|\theta'^m\right)|_{\mathcal{A}'^m_{t+1}=\pi'^m(\mathcal{S}^m_{t+1})}, \quad (12)$$

[2]During training, ($\mathcal{S}^{\text{All}}$, $\mathcal{A}^{\text{All}}$) sharing needs $2(N+K)$ sets for information synchronization, because there are K users and N SBSs agents in the system, and each agent has F-dimensional action and state.

TABLE I: Neural Network Configuration of MADDPG

Agent	Name	Neurons Num. and Active Fun	Type
Users/ SBSs	Input	F for $\mathcal{S}_t^m$, ReLU	Actor
	Hidden	2 layers with 300 neurons for each , ReLU	
	Output	F for $\mathcal{A}_t^m$, Sigmoid	
	Input	$2(N+K)F$ † for ($\mathcal{S}^{\mathrm{All}}$, $\mathcal{A}^{\mathrm{All}}$), ReLU	Critic
	Hidden	2 layers with 1024 neurons for each, ReLU	
	Output	1 ‡,NA	

† For both users and SBSs, the critic input is the action-state pair ($\mathcal{S}^{\mathrm{All}}$, $\mathcal{A}^{\mathrm{All}}$) with the dimension $2(N+K)F$.

‡ The critic output is the policy gradient value scaler, so the output dimension of critic are all 1.

where $\mathcal{A}'^{\mathrm{All}}_{t+1}$ is the collection of global action sampled from the mini-batch. So there is $\mathcal{A}'^{\mathrm{All}}_{t+1} = [\mathcal{A}'^{1}_{t+1}, \mathcal{A}'^{m}_{t+1}, \cdots, \mathcal{A}'^{M}_{t+1}]$. The critic online network also iteratively updates the policy gradient with the input of actor online network.

$$\nabla_{\theta^\mu} J(\mu) = \nabla_a Q(\mathcal{S}^{\mathrm{All}}, \mathcal{A}^{\mathrm{All}}|\theta^m)|_{\mathcal{S}^{\mathrm{All}}=\mathcal{S}_t^{\mathrm{All}}, \mathcal{A}^{\mathrm{All}}=[\pi^1(\mathcal{S}_t^1|\mu^1),\cdots,\pi^M(\mathcal{S}_t^M|\mu^M)]} \times \nabla_\mu \pi^m(\mathcal{S}^m|\mu^m). \quad (13)$$

- Critic target $Q'^m(\mathcal{S}^{All}, \mathcal{A}'^{All}|\theta'^m)$: updates the target critic value y_t^m with the inputs of $\mathcal{S}_{t+1}^{\mathrm{All}}$ and $\mathcal{A}'^{\mathrm{All}}_{t+1}$.

For both the actor and critic target networks, the parameters of each agent are soft updated as:

$$\begin{cases} \mu'^m \leftarrow \tau\mu^m + (1-\tau)\mu'^m, \\ \theta'^m \leftarrow \tau\theta^m + (1-\tau)\theta'^m, \end{cases} \quad (14)$$

where τ is the forgetting factor.

IV. Numerical Results

A. Simulation Configuration

Supposing that there are $N = 5$ SBSs and $K = 3$ users in the CRN system. All the SBSs are geographically distributed nearby users in the radius of 600m but more than 1000m from PBS. The transmission power of PBS and SBSs are 1W and 40mW respectively, and the noise power is $\delta^2 = -174$dBm/Hz. Thus $r_{0,k}$ is nearly only one third of $r_{n,k}$. The time slot is $T_0 = 100$s and the available bandwidth is $B = 5$MHz.

There are $F = 50$ files following Pareto distribution with the shape parameter 1.2 and the average size $\bar{s} = 1$Mbits. The contents request times $q_{k,f}$ at each user submits to Zipf distribution [2] with the shape parameter 0.8[3] and the average value $\bar{q}_{k,f} = 20$.

The incentive price is $\epsilon = 0.1$ and the replacement coefficient is $\beta = 1.0$. The storage capacity of SBSs is set to be $C = 8$Mbits uniformly. The neural network configuration is listed in Table I and RB is $\mathbb{M} = 10$Kbits.

B. Effectiveness of Heterogeneous MADDPG

The location of users, SBSs and PBS is depicted in Fig.2(a). The reward of users and SBSs is converged along with the training episodes in Fig.2(b) and Fig.2(c) respectively, which proves the effectiveness of the proposed heterogeneous MADDPG algorithm.

Fig.3 further evaluates the impact of location and replacement cost to the reward and storage strategy of SBSs respectively. The shadow enclosed by each curve denotes that the storage capacity is limited to C.

- Fig.3(a) implies that the reward of SBSs is related to their geographic distribution. In specific, the 2th and 4th SBS, who are farther from users, show a worse reward (in Fig.2(c)) and different popularity-fraction curves (in Fig.3(a)) compared with other SBSs. That is because they have to store the hottest contents as many as possible to attract more users, which will sacrifice some contents with less popularity. This less-than-ideal cache strategy overcomes the drawback of worse transmission rate and leads to a better reward.
- Fig.3(a) also depicts that the cached fraction is changing along with the contents popularity which is normalized to $\sum_{f\in\mathcal{F}} p_{n,f} = 1$. Statistically, a more popular content will be stored by a larger fraction.
- Fig.3(b) demonstrates the impact of replacement cost on the storage strategy[4]. As β in Eq.(8) increases from 0.5 to 1.2, the popularity-fraction curves change from up-concave to down-convex, which means SBSs tend to cache more popular contents in coherent with model's expectation. In specific, when the cache space becomes rare (caused either by low-rate or high replacement cost), the storage will used by hot contents in priority.

C. Performance Comparison with Homogeneous MADRL

Fig.4 compares the overall reward of users ($\sum_{k=1}^{K} \mathcal{R}_t^k$) and SBSs ($\sum_{n=1}^{N} \mathcal{R}_t^n$) respectively in three different scenarios.

- Red lines denote the reward that only SBSs learn to optimize the storage without considering users' action. Users adopt the static fetching strategy, i.e. always fetch the largest fraction of requested content from SBSs. At this time, the heterogeneous MADDPG degrades to homogeneous MADDPG.
- Green lines denote the reward that only users learn to select the best SBSs for the minimum delivery delay, while SBSs adopt the least frequently used (LFU) replacement strategy[5].
- Blue lines are the reward of the proposed heterogeneous MADDPG, where both users and SBSs learn the contents fetching and caching storage strategy distributively.

The reward converges along with episodes in all three scenarios, which proves again that multi-agent DRL is an effective way for multi-cache optimization problem. However, whatever the reward of users or SBSs, the heterogeneous MADDPG is superior to those homogeneous learning approach, e.g. the total reward ($\sum_{k=1}^{K} \mathcal{R}_t^k + \sum_{n=1}^{N} \mathcal{R}_t^n$) can be improved by nearly 40%. That is because heterogeneous agents can autonomously learn to balance the mixed cooperative-competitive relationship for a lower delivery delay and higher cache efficiency.

D. Performance Comparison with Other DRL

Other actor-critic based off-policy DRL algorithms, including DDPG, proximal policy optimization [14] (PPO) and twin delay DDPG [15] (TD3) are compared with the proposed heterogeneous MADDPG in Fig.5. They are designed for

[3] Over 80% requests aim to the most popular 20% contents

[4] Since all SBSs present similar features, we only depict the 1th SBS as an example.

[5] Least recently used (LRU) replacement strategy is tested with only a slight performance loss compared with LFU in this scenario.

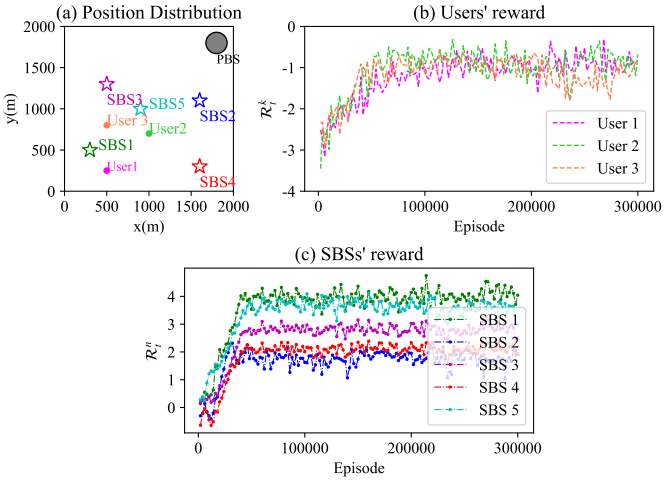

Fig. 2: Locations and rewards of heterogeneous agents

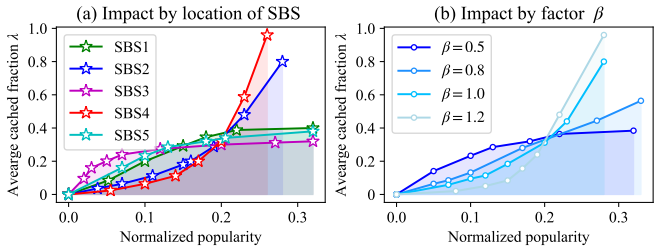

Fig. 3: Contents popularity vs. cached fraction

a single agent which leads to a scalar global reward [1]. Thus, to facilitate the comparison, the value of global reward $\sum_{k=1}^{K}\mathcal{R}_t^k+\sum_{n=1}^{N}\mathcal{R}_t^n$, as well as the corresponding variance are used for performance comparison.

Obviously, MADDPG outperforms all the other DRL algorithms. The reason is that it can stimulate users to fetch from SBSs leading to a lower delivery delay, while prompt SBSs to optimize their storage for a better revenue simultaneously. Meanwhile, when cooperators and competitors co-exist in the environment, the heterogeneous MADDPG helps the agents to converge for a better global reward.

V. Conclusion

Deploying distributed cooperative cache in CRNs, can decrease the traffic loads, reduce the delivery delay, and free up the spectrum resource for other users. However, the contents requirements are hard to predict. Besides, the cooperation and competition among users and SBSs makes the optimized problem more complex to be solved. DRL provides an effective way to tackle the no-trivial issue. Unlike other DRL approaches that only aim at learning either the storage optimization of SBSs or the fetching action of users, the paper proposes a heterogeneous MADDPG algorithm, which novelly features at treating users and cache nodes as two types of heterogeneous agents to learn how to cooperate and compete spontaneously. The numeral results demonstrate that comparing with other DRL approaches like homogeneous MADRL, PPO, DDPG and TD3, the proposed MADDPG can enhance the system reward by optimizing the content storage of SBSs and delivery delay of users simultaneously.

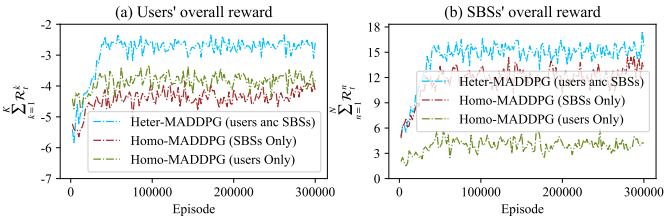

Fig. 4: Comparison with homogeneous MADDPG

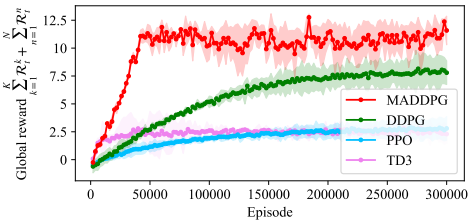

Fig. 5: Performance comparison with other DRL

References

[1] Z. Zhang and M. Tao, "Deep learning for wireless coded caching with unknown and time-variant content popularity," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1152–1163, 2021.

[2] B. Zhou, S. Wang, E. Chen, and M. Tao, "Cooperative caching for spectrum access in cognitive radio networks," in *2017 IEEE International Conference on Communications (ICC)*, Paris France, May 2017.

[3] D. Das and A. A. Abouzeid, "Co-operative caching in dynamic shared spectrum networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 7, pp. 5060–5075, 2016.

[4] P. Wu, J. Li, L. Shi, M. Ding, K. Cai, and F. Yang, "Dynamic content update for wireless edge caching via deep reinforcement learning," *IEEE Commun. Lett.*, vol. 23, no. 10, pp. 1773–1777, 2019.

[5] C. Zhong, M. C. Gursoy, and S. Velipasalar, "Deep reinforcement learning-based edge caching in wireless networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 1, pp. 48–61, 2020.

[6] X. Wu, J. Li, M. Xiao, P. C. Ching, and H. Vincent Poor, "Multi-agent reinforcement learning for cooperative coded caching via homotopy optimization," *IEEE Trans. Wireless Commun.*, vol. 7, no. 4, pp. 1–1, 2021.

[7] J. Du, C. Jiang, E. Gelenbe, H. Zhang, Y. Ren, and T. Q. S. Quek, "Double auction mechanism design for video caching in heterogeneous ultra-dense networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 3, pp. 1669–1683, 2019.

[8] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, "Learning distributed caching strategies in small cell networks," in *IEEE Int. Symp. on Wireless Commun. Sys. (ISWCS)*, Bologna, Italy, Aug. 2014.

[9] J. Liao, K.-K. Wong, Y. Zhang, Z. Zheng, and K. Yang, "Coding, multicast, and cooperation for cache- enabled heterogeneous small cell networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 10, pp. 6838–6853, 2017.

[10] J. Yang, H. Xu, and J. Zhang, "Exploiting secondary caching for cooperative cognitive radio networks," *IEEE Commn. Lett.*, vol. 23, no. 1, pp. 124–127, 2019.

[11] L. Shi, L. Zhao, G. Zheng, Z. Han, and Y. Ye, "Incentive design for cache-enabled d2d underlaid cellular networks using stackelberg game," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 765–779, 2019.

[12] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Pro. of Int. Conf. on Neural Information Processing Systems (NIPS)*, Long Beach, Dec. 2017.

[13] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Puerto Rio, May 2016.

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," in *Proc. of Machine Learning Research (PMLR)*, Edinburgh Scotland UK, June 2017.

[15] S. Fujimoto, H. V. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. of Machine Learning Research (PMLR)*, Stockholm Sweden, June 2018.