Reinforcement Learning Based Cooperative Coded Caching under Dynamic Popularities in Ultra-Dense Networks

Shen Gao, Peihao Dong, *Student Member, IEEE*, Zhiwen Pan, *Member, IEEE*, and Geoffrey Ye Li, *Fellow, IEEE*

Abstract—For ultra-dense networks with wireless backhaul. caching strategy at small base stations (SBSs), usually with limited storage, is critical to meet massive high data rate requests. Since the content popularity profile varies with time in an unknown way, we exploit reinforcement learning (RL) to design a cooperative caching strategy with maximum-distance separable (MDS) coding. We model the MDS coding based cooperative caching as a Markov decision process to capture the popularity dynamics and maximize the long-term expected cumulative traffic load served directly by the SBSs without accessing the macro base station. For the formulated problem, we first find the optimal solution for a small-scale system by embedding the cooperative MDS coding into Q-learning. To cope with the large-scale case, we approximate the state-action value function heuristically. The approximated function includes only a small number of learnable parameters and enables us to propose a fast and efficient action-selection approach, which dramatically reduces the complexity. Numerical results verify the optimality/near-optimality of the proposed RL based algorithms and show the superiority compared with the baseline schemes. They also exhibit good robustness to different environments.

Index Terms—Ultra-dense network, reinforcement learning, cooperative coded caching, popularity dynamics.

I. INTRODUCTION

N recent years, the mobile data traffic increases dramatically and overwhelms the current fourth generation (4G) systems [1]. As one of the key technologies in the fifth generation (5G) systems, ultra-dense network (UDN) can improve the system throughput significantly by deploying multiple small base stations (SBSs) that coexist with the macro base stations (MBSs) [2], [3]. By exploiting efficient interference coordination [4], UDN can improve the system throughput per unit area (on the scale of km²) almost linearly with the number of SBSs. Wireless backhaul technology [5], [6] is

The work of S. Gao and Z. Pan was supported by National Key Research and Development Project under Grant 2018YFB1802402 and 333 Program of Jiangsu under Grant BRA2017366. The work of S. Gao was also supported by China Scholarship Council (CSC) under Grant 201706090065. (Corresponding author: Zhiwen Pan, Peihao Dong.)

S. Gao and Z. Pan are with the National Mobile Communications Research Laboratory, Southeast University, Nanjing, Jiangsu 210096, China, and are also with the Purple Mountain Laboratories, Nanjing, Jiangsu 211100, China (e-mail: gaoshen@seu.edu.cn; pzw@seu.edu.cn).

P. Dong is with the National Mobile Communications Research Laboratory, Southeast University, Nanjing, Jiangsu 210096, China (e-mail: phdong@seu.edu.cn).

G. Y. Li is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: liye@ece.gatech.edu).

regarded as a feasible solution to overcome the installation obstacle to wired backhaul caused by the expensive costs and hard-to-reach locations of SBSs. However, the need to forward massive data traffic poses as a main challenge for wireless backhaul due to its limited spectrum resources. One thus has to resort to smart content caching at the edge of the network to alleviate backhaul congestion so as to afford satisfactory quality of experience (QoE) [7], [8]. In [9], it is shown that the data traffic can be decreased by 45% through proper caching. Nonetheless, the limited storage capacity at each SBS crucially calls for proper prioritization of content for caching at the cell edge to best meet user requests. How to design efficient caching strategies has thus drawn much research interest. Recently, various trials driven by machine learning are widely emerged to optimize communication networks and have achieved great success [10]-[17]. Therefore, reinforcement learning (RL) may be a promising solution for caching design compared to the traditional optimization based methods.

A. Related work

The authors in [18]–[24] have designed the non-cooperative caching strategy, where the user fetches the requested full content from the BS. A joint design of caching and routing policies in [18] is formulated under hard bandwidth constraints of the SBSs and optimized based on the facility locations. To cope with the limited cache space, the caching strategy in [19] is optimized based on multicast transmission by using randomized-rounding techniques. Based on diversity transmission, a probabilistic caching strategy is proposed in [20] to minimize the content delivery latency in spatially clustered networks. In [21], a joint caching and multicasting approach is designed for large-scale heterogeneous networks (HetNet), based on which the successful transmission probability is derived and optimized to provide the best performance. In [22], the average delivery rate is analyzed for a two-tier HetNet with inter-tier and intra-tier dependence, where the most popular contents are cached at the SBSs. The tier-level content placement is investigated in [23] to maximize the hit probability for a multi-tier HetNet. A cache-based content delivery scheme is proposed in [24] in terms of the ergodic rate, outage probability, throughput, and delay for a three-tier HetNet including BSs, relays, and device-to-device pairs.

In contrast to the non-cooperative caching strategy simply storing full contents, it is preferable for the BSs to store the fragments of contents and serve a common user in a cooperative manner, which reduces the backhaul overhead significantly but at the cost of relatively high computational complexity. In [25], a coded scheme is compared with a non-cooperative scheme and is shown to be superior in terms of the expected downloading delay. Two fundamental metrics, the expected backhaul rate and the energy consumption, are minimized in [26] for maximum-distance separable (MDS) coding based cooperative caching scheme in a two-tier HetNet. In [27], a combined caching scheme is developed, where a reserved part of the cache space is used for cooperative caching at the SBSs.

The above work utilizes stochastic geometry and conventional optimization algorithms to optimize caching strategy based on the acquired popularity profile, which may cause performance degradation if the popularity profile evolves unpredictably with time. Fortunately, RL has been shown to perform well in sequential decision making by capturing the unknown and nondeterministic environmental dynamics for use in various caching problems in wireless networks. To minimize the long-term average energy cost, a thresholdbased proactive caching strategy is proposed in [28] for cache-enabled mobile users, where RL algorithms are used to optimize the parameters representing the threshold values. In [29], the cooperative transmission among local caches is exploited to achieve a trade-off between the hit ratio and the average delay based on a multi-agent RL framework, where a user will resort to other local caches when the content is unavailable at its associated local cache. The grouped linear model is introduced in [30] to obtain the predicted content requests, based on which the cache replacement is optimized by using RL with model-free acceleration. In [31], a RL framework is proposed to obtain the optimal caching strategy at SBSs taking into account the space-time dynamics of the content popularity. In [32], the probabilistic caching strategy, resource allocation, and computation offloading at fog nodes are jointly considered to minimize the average transmission delay exploiting deep RL. In [28]–[32], only full contents, instead of the fragments, are stored. As we can see in our subsequent work in this article, the power of RL can be sufficiently exploited by storing only part of contents.

B. Motivation and Contribution

In [18]—[27], caching strategies are designed by assuming a time-invariant content popularity profile, which may not yield desirable performance when facing the dynamic popularity profile in the real scenario. In contrast, we exploit RL to capture the dynamics of content popularity evolution through interactions with the environment. The RL based solution incorporates the inherent characteristics of the content popularity profile and popularity transition and thus is suitable for the practical scenario with changeable popularity profile. On the other hand, different from full content caching in the traditional network architecture [28]—[32], we introduce the cooperative caching strategy storing coded fragments at SBSs so that the limited storage can be utilized more efficiently for UDN. The feasibility is guaranteed by the fact that the UDN

architecture enables a user to be served by more than one SBS. Moreover, the significantly increasing SBSs in UDN can contribute much more unbiased data to the global popularity to help the MBS improve the RL policy and to enhance the effectiveness of the cooperative coded caching. Although the cooperative coded caching is more efficient, it poses daunting challenges when formulating the RL based problem in such complex UDN. In this paper, we design the cooperative coded caching strategy for UDN exploiting RL and the main contributions can be summarized as follows.

- 1) Prior works either assume the time-invariant content popularity profile or use the less efficient full content caching. We develop a cooperative coded caching strategy using RL algorithms, which is a promising solution for backhaul offloading in UDN. Under an unknown dynamic environment on usage traffic, we figure out how to cooperate among the SBSs and what kind of coding method to choose to stimulate the potential of the SBS cooperation most. Afterwards, we abstract the MDS coding based cooperative caching strategy design into a RL based sequential decision making problem, which is mathematically modeled as a Markov decision process (MDP) to maximize the long-term expected cumulative traffic load served directly by the SBSs without accessing the MBS. Our formulated problem matches the real scenario well and can reduce the performance loss caused by popularity profile mismatch.
- 2) For the formulated RL based problem, we successfully find the matched solution for performance maximization. After evaluating the feasibility, performance, and complexity of the possible RL algorithms comprehensively, we propose to embed the complicated cooperative MDS coding into the Q-learning based solution. The developed Q-learning based cooperative coded caching strategy is shown optimal for the formulated RL problem with an acceptable complexity in the small-scale system.
- 3) As the system dimensionality becomes large, the Q-learning based algorithm will malfunction and finding the appropriate solution is challenging due to the prohibitively huge action space. To address this problem, we first approximate the state-action value function heuristically according to the instantaneous reward. The approximated state-action value function includes only a small number of learnable parameters and enables us to propose a fast and efficient action-selection approach, which dramatically reduces the complexity. The developed value function approximation based algorithm can be flexibly applied to the large-scale system yet still yielding near-optimal performance.

The rest of the paper is organized as follows. The considered system model is described in Section II. Section III presents the RL based problem formulation and proposes a Q-learning based solution for a small-scale cooperative coded caching system. The large-scale solution is further developed in Section IV by using value function approximation. Numerical results are presented in Section V to verify our proposed solutions. Finally, Section VI provides concluding remarks.

II. SYSTEM MODEL

In this section, we first give an overview of the network model, followed by the description of the content popularity profile. Then we introduce the MDS coded cooperative caching, based on which the cooperative transmission among BSs is provided.

A. Network Model

Consider the downlink of a two-tier UDN with an MBS and p SBSs as illustrated in Fig. 1. In the coverage area of the MBS, p SBSs are deployed to provide better coverage and enhance system throughput. $\mathcal{G} = \{\mathcal{G}_m, \mathcal{G}_s\}$ is the set of users in the network, where \mathcal{G}_m and \mathcal{G}_s represent the subsets including users that are served by the MBS directly and that connect to the MBS through a SBS, respectively, and can be specified by load balancing [33]. The links between users and BSs, i.e., MBS and SBSs, are called radio access links and those connecting SBSs and the MBS are referred to as wireless backhaul links. In addition, the MBS can retrieve all contents from the content provider through the core network router and optical fibers. The radio access links and wireless backhaul links operate on orthogonal spectra, that is, the whole spectrum B_w is divided into two parts denoted as B_{al} and $B_w - B_{al}$ for the radio access links and the wireless backhaul links, respectively, as in [34], [35]. Densely-deployed SBSs bring a daunting challenge to wireless backhaul links when serving massive concurrent user requests. Users may suffer from unbearable delays, which result in a bad user experience due to the congestion of wireless backhaul links during the peak hours of transmission. Therefore, caches, which are usually with limited storage capacity, are deployed at SBSs to offload the wireless backhaul load and reduce the queueing delay so that QoE can be improved for users. The SBSs cache contents to serve the users in \mathcal{G}_s , which consequently will be focused on hereinafter. Without loss of generality, we assume that no transmission error occurs when any SBS delivers contents to the users in its coverage range. Since the users in \mathcal{G}_s can only be served by the SBSs, the transmission error will impact the performance similarly no matter whether the SBSs cache the contents or not.

The considered heterogeneous UDN model abstracted from [36] has been recognized as the solution for the future network architecture and widely investigated in lots of studies [37]-[41]. It is noted that except the well-known effect on improving the system throughput significantly, the denselydeployed SBSs can also facilitate the RL based cooperative coded caching design. Specifically, the densely-deployed SBSs can collect and report sufficient user requests to the MBS. Only in this case, the user requests received by the MBS can reflect the actual user request distribution unbiasedly and accurately according to the law of large numbers. Then the MBS is able to design a more efficient caching strategy that works well in the real scenarios by using RL algorithms. Furthermore, the high density of SBSs means that a user can be served by more SBSs and thus UDN is more favorable for the cooperative coded caching compared to the traditional network architecture.

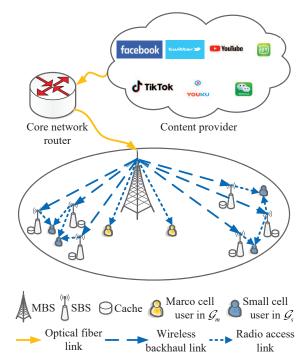


Fig. 1. Network model.

B. Content Popularity Profile

Users in the network request contents from a set denoted by $\mathcal{C} = \{1, 2, \dots, C\}$, where the size of each content is equal to B [26]. According to the alternation of peak and off-peak hours, the whole transmission time is slotted and each time slot t contains peak hours and off-peak hours of transmission. Denote $\mathbf{N}_i(t) = [N_{i1}(t), N_{i2}(t), \dots, N_{iC}(t)]$ as the vector composed of the number of user requests for each content during the peak hours of time slot t collected by the ith SBS for $i = 1, 2, \dots, p$. $\mathbf{N}(t) = [N_1(t), N_2(t), \dots, N_C(t)]$ is the aggregation of $\mathbf{N}_1(t), \mathbf{N}_2(t), \dots, \mathbf{N}_p(t)$ at the MBS. $\boldsymbol{\theta}(t) = [\theta_1(t), \theta_2(t), \dots, \theta_C(t)]$ denotes the corresponding content popularity vector, defined as

$$\theta_c(t) = \frac{N_c(t)}{\sum_{j=1}^C N_j(t)}, \quad c \in \mathcal{C}.$$
 (1)

The user requests used to compute the content popularity profile exhibit the specific distribution. The user request distribution has been studied in [42] and [43] by using the real traces of user requests and modeled as a Zipf-like distribution. Specifically, six traces of the web proxies are collected in [42] from academic, corporate, and Internet service provider environments and prove to follow Zipf-like behaviors. Authors in [43] investigate the popular wireless mobile network based

¹Note that, the approach in this paper can also be applied to the case of different content sizes. Specifically, contents with different sizes can be properly combined or divided into pieces with the same size to guarantee the equal size for each content.

²In a time slot, peak hours refer to the period when the network experiences high traffic load with numerous requests generated. The following off-peak hours represent the period when the content requests from users are relatively inactive. The duration of peak hours is usually longer than that of off-peak hours. In the practical scenario, the network operator may adjust the length of the time slots and the durations of peak and off-peak hours dynamically according to the traffic load.

on a mass of representative data from the telecom operator and demonstrate the good fitness of the Zipf-like distribution. It is noted that users in an area generally exhibit diverse individual preferences for the contents while the overall user requests of this area aggregating the individual preference of each user follow the Zipf-like distribution. According to [44], the skewness of the Zipf-like distribution is dependent on the specific application. In brief, Zipf-like distribution is able to well depict the real user request distribution of various networks and thus is widely adopted in [18], [21], [25], [26], [31]. In this paper, we consider the mobile UDN and thus Zipf-like distribution can be safely used to model the user request distribution. As it will be mentioned in Section V, the simulation data are generated following Zipf-like distribution with the skewness set according to [44].³

C. MDS Coded Cooperative Caching

From [25], storing fragments of contents, instead of complete contents, in caches actually performs better in offloading the wireless backhaul. In the case of storing uncoded fragments, a request from a user will be met only when the user collects all content fragments. In contrast to simply splitting a content into multiple coarse fragments, utilizing coded caching scheme can further improve the performance. For MDS coding, B information bits are coded as a string of parity bits with an arbitrary length and then split into multiple packets. According to [25], a user can recover the full content so long as it collects a certain number of packets with the total size no less than B parity bits since the individual identity of each bit is irrelevant. Thus, caching design at the SBSs matters since it determines how many parity bits the SBSs can deliver directly to the user without resorting to the MBS. A cache memory with a limited size of $K \times B$ bits is installed at each SBS. Let $\mathbf{a}(t) = [a_1(t), a_2(t), \dots, a_C(t)]$ denote the SBSs caching decision vector designed by the MBS during the off-peak hours of time slot t, where $a_c(t) \in [0,1]$ is the normalized fraction of content c cached at the SBSs. The elements in $\mathbf{a}(t)$ are subject to the constraint $\sum_{i=1}^{C} a_i(t) = K$ and the set of stored contents with $a_c(t) \ge a_c(t-1)$ is denoted by $\mathcal{C}^{'}(t)$. The MBS converts the content $c \, (c \in \mathcal{C}^{'}(t))$ into $B_c^{\rm MDS}$ parity bits expressed as

$$B_c^{\text{MDS}} = (p+1)B, \quad c \in \mathcal{C}'(t). \tag{2}$$

The B_c^{MDS} parity bits are divided into two non-overlapped candidate sets, $\mathcal{B}_c^{\text{SBS}}$ and $\mathcal{B}_c^{\text{MBS}}$, respectively. $\mathcal{B}_c^{\text{SBS}}$ contains pB bits and is equally divided into p non-overlapped candidate sets, $\mathcal{B}_c^1, \ldots, \mathcal{B}_c^p$, for p SBSs, respectively. $\mathcal{B}_c^{\text{MBS}}$ including the remaining B bits is the candidate set for the MBS.⁴ Then the MBS delivers $(a_c(t) - a_c(t-1))B$ bits from each of $\mathcal{B}_c^1, \ldots, \mathcal{B}_c^p$ to the corresponding SBS for caching. Assuming there are d SBSs serving the user simultaneously, the MBS

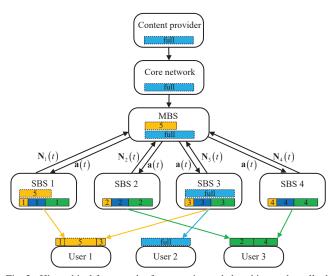


Fig. 2. Hierarchical framework of cooperative coded caching at the cell edge. Different contents are distinguished by different colors. The indexed rectangles with the same color denote the different coded fragments of a content, where the solid and dashed lines indicate that the fragments are stored in the SBSs and the MBS, respectively. The rectangle with the dashed line and label "full" represents the full content unavailable at any of SBSs that can cover the user and fetched from the content provider.

should preserve arbitrary $(1 - da_c(t))B$ bits from $\mathcal{B}_c^{\text{MBS}}$ when $da_c(t) \leq 1$.

D. BSs Cooperative Transmission Scheme

In UDN, densely-deployed SBSs enable users to be served by multiple SBSs simultaneously. Moreover, MDS coded caching at each SBS makes the cooperative transmission more effective. User $g \in \mathcal{G}_s$ selects d SBSs providing the strongest reference signal received power (RSRP) to form its cooperative serving set \mathcal{S}_g^d . When user g requests content e during the peak hours of time slot e 1 and e 2, it will receive e 2 coded bits from e 3 associated SBSs. If e 3 the user is able to restore the full content e directly. Otherwise, the MBS chooses a SBS e 3 with the strongest RSRP from e 3 and sends the complementary e 1 the strongest RSRP from e 4 and sends the complementary e 3. In the case of e 6, the MBS transmits the original content with the size of e bits to e via the SBS e 3.

To provide a more comprehensive illustration on how this network works, Fig. 2 shows the hierarchical framework of cooperative coded caching abstracted from Fig. 1. All SBSs report their respective observations on the numbers of content requests to the MBS and the latter makes caching decision for them, which leads to three possible service modes for different content requests. User 3 receives coded fragments 2 and 4, both with the size of $\frac{B}{2}$ bits of the desired content from two different SBSs, and then performs recovery. In contrast, user 1 only collects coded fragments 1 and 3 with the size of $\frac{B}{6}$ bits, which are insufficient to recover the desired content. To fill the rest of the content, the MBS sends the complementary coded fragment 5 with the size of $\frac{2B}{3}$ bits to user 1 through SBS 1. In addition, user 2 requests a content unavailable at any of SBSs that can cover it, i.e., the rectangle with the label "full" in Fig. 2, and the MBS sends the full content with the size of B bits to user 2 through SBS 3.

³It is noted that the RL based approach is designed in a model-free manner and its ability to capture the inherent popularity evolution is independent of any specific distribution model according to Section III and IV. To verify the effectiveness of the proposed approach, we employ the practical Zipf-like distribution to generate the simulation data.

 $^{{}^{4}\}mathcal{B}_{c}^{\text{MBS}},\,\mathcal{B}_{c}^{\text{SBS}}$, and $\mathcal{B}_{c}^{1},\ldots,\mathcal{B}_{c}^{p}$ are constant for all time slots.

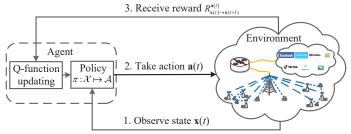


Fig. 3. The agent-environment interactions in a reinforcement learning problem.

III. REINFORCEMENT LEARNING BASED COOPERATIVE CODED CACHING

In this section, the elements in the transition tuple of MDP are first specified in the cooperative coded caching scenario, based on which the corresponding objective function is formulated. Then a Q-learning based cooperative coded caching strategy that provides optimal solution is proposed to solve the formulated problem in a small-scale system.

A. Reinforcement Learning based Problem Formulation

As shown in Fig. 3, the agent including the MBS and the SBSs in its coverage interacts with the environment that is regarded as everything outside the agent, where the MBS makes the decision and then the SBSs execute this decision. The process can be abstracted into a RL problem, which can be modeled as a MDP and expressed as a transition tuple $\left(\mathcal{X}, \mathcal{A}, P_{\mathbf{x}(t) \to \mathbf{x}(t+1)}^{\mathbf{a}(t)}, R_{\mathbf{x}(t) \to \mathbf{x}(t+1)}^{\mathbf{a}(t)}\right)$. At time slot t, the agent in the environment observes a state $\mathbf{x}(t)$, which belongs to the state space \mathcal{X} , and then takes an action $\mathbf{a}(t)$ chosen from the action space \mathcal{A} according to the policy $\pi: \mathcal{X} \mapsto \mathcal{A}$. After that, the environment evolves from the current state $\mathbf{x}(t)$ to a new one, $\mathbf{x}(t+1)$, in a transition probability $P_{\mathbf{x}(t) \to \mathbf{x}(t+1)}^{\mathbf{a}(t)}$ and feeds back the reward $R_{\mathbf{x}(t) \to \mathbf{x}(t+1)}^{\mathbf{a}(t)}$ to the agent. The policy π needs to be updated at each time slot until obtaining the optimal state-action value function, called Q-function. The reward received from the environment is used to update the Q-function and prepare for making decision in the following slot. To be specific, key elements of the RL based cooperative coded caching design are described as follows.

- 1) State: As the characterization of the environment, the state is unknown to the agent and needs to be acquired through the observation. During the off-peak hours of time slot t, the observation result includes two parts: the numbers of requests for all contents during the peak hours of time slot t, $\mathbf{N}(t)$, and the caching decision made during the off-peak hours of time slot t-1, $\mathbf{a}(t-1)$, which are necessary for the selection of $\mathbf{a}(t)$. Furthermore, $\mathbf{N}(t)$ is normalized as $\boldsymbol{\theta}(t)$ according to (1) to acquire the underlying transition mode of content requests. Therefore, the state at time slot t can be expressed as $\mathbf{x}(t) = [\boldsymbol{\theta}(t), \mathbf{a}(t-1)]$.
- 2) Action: Based on the observed state $\mathbf{x}(t)$ and the updated policy π , the agent will decide which and how many contents should be stored at the SBSs during the off-peak hours of time slot t. Hence, the action is denoted as the caching decision

vector $\mathbf{a}(t) = [a_1(t), a_2(t), \dots, a_C(t)], \text{ where } \sum_{i=1}^C a_i(t) = K$

- 3) Transition Probability: Following the action $\mathbf{a}(t)$, the environment state transits to the next state $\mathbf{x}(t+1)$ at the end of peak hours of time slot t+1 with probability $P_{\mathbf{x}(t) \to \mathbf{x}(t+1)}^{\mathbf{a}(t)}$. It is usually determined by the environment, independent of the agent.
- 4) Reward Design: The agent will receive a reward when the environment state transits to $\mathbf{x}(t+1)$. The system performance can be enhanced when the designed reward at each time slot correlates with the desired goal. In the studied cooperative coded caching problem, our goal is to maximize the total traffic load served directly by the SBSs without accessing the MBS over all the time slots, which is consistent with the general aim of RL that maximizes the expected cumulative discounted rewards. Therefore, the received reward at time slot t+1 is set as

$$R_{\mathbf{x}(t)\to\mathbf{x}(t+1)}^{\mathbf{a}(t)} = \sum_{i=1}^{C} N_i(t+1) - p \sum_{i=1}^{C} \max(a_i(t) - a_i(t-1), 0)$$
$$-\sum_{i=1}^{C} N_i(t+1) \sum_{j=1}^{C} \theta_j(t+1) \max(1 - da_j(t), 0),$$
(3)

where the first term denotes the total traffic load, i.e., the total number of content requests from users, during the peak hours at time slot t+1, the second term represents the traffic load of updating the contents cached in the SBSs during the off-peak hours at time slot t, and the third term accounts for the traffic load of transmitting the complementary coded fragments from the MBS to the SBSs during the peak hours at time slot t+1, respectively.⁵ Please note that there are also some other signaling overheads in the agent-environment interaction process, including the overhead for reporting the user request vectors, $\mathbf{N}_i(t), \forall i \in \{1, \dots, p\}$, from all SBSs to the MBS and the overhead for broadcasting the action vector, $\mathbf{a}(t)$, in reverse, where each element in $\mathbf{N}_i(t)$ and $\mathbf{a}(t)$ can be represented by only tens of bits or less. Thus, these signaling overheads are negligible compared to the actual traffic loads corresponding to the second and third terms in (3) and can be omitted safely.

Note that it is reasonable to formulate this cooperative coded caching problem as an MDP and the feasibility is revealed by the abstracted graphical model of caching process shown in Fig. 4. From the figure: i) $\mathbf{x}(t+1)$ only depends on $\mathbf{x}(t)$ but is independent of $\mathbf{x}(t-1), \mathbf{x}(t-2), \ldots$ ii) $\mathbf{a}(t)$, which is taken based on $\mathbf{x}(t)$, determines $\mathbf{x}(t+1)$ with the transition probability $P_{\mathbf{x}(t) \to \mathbf{x}(t+1)}^{\mathbf{a}(t)}$. Denoting $\mathbf{x}(t)$, $\mathbf{a}(t)$, and $\mathbf{x}(t+1)$ by \mathbf{x} , \mathbf{a} , and \mathbf{x}' , respectively, the transition probability satisfies $\sum_{\mathbf{x}' \in \mathcal{X}} P_{\mathbf{x} \to \mathbf{x}'}^{\mathbf{a}} = 1$. If the MBS makes another caching decision, $\bar{\mathbf{a}}$, based on \mathbf{x} , then the transition probability satisfies $P_{\mathbf{x} \to \mathbf{x}'}^{\mathbf{a}} \neq P_{\mathbf{x} \to \mathbf{x}'}^{\bar{\mathbf{a}}}$. iii) $R_{\mathbf{x}(t) \to \mathbf{x}(t+1)}^{\mathbf{a}(t)}$ is determined by three types of traffic loads in (3), which are dependent on $\mathbf{x}(t)$, $\mathbf{x}(t+1)$, and $\mathbf{a}(t)$. Therefore, the caching process satisfies the MDP properties.

 $^{^5{\}rm The}$ three terms in (3) are normalized traffic loads obtained by dividing the corresponding actual traffic load by B.

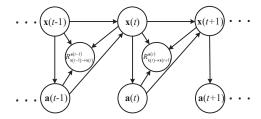


Fig. 4. Graphical model of the caching process.

5) Objective Function: The objective of RL is to maximize the expected cumulative discounted rewards over the transition probability, which is expressed as [45]

$$G_{\gamma}^{\pi}(t) = \mathbb{E}\left[\sum_{n=0}^{\infty} \gamma^n R_{t+n}\right],$$
 (4)

where $\mathbb{E}[\cdot]$ represents the expectation, $\gamma \in [0,1]$ is the discounted factor, and R_{t+n} denotes the successive received reward from the state $\mathbf{x}(t)$.

Use the Q-function $Q_{\gamma}^{\pi}(\mathbf{x}, \mathbf{a})$ to denote the expected cumulative discounted rewards taking an arbitrary action \mathbf{a} under the current state \mathbf{x} and then following the policy π . Thus, $Q_{\gamma}^{\pi}(\mathbf{x}, \mathbf{a})$ is given by

$$Q_{\gamma}^{\pi}(\mathbf{x}, \mathbf{a}) = \mathbb{E}\left[\sum_{n=0}^{\infty} \gamma^{n} R_{t+n} | \mathbf{x}(t) = \mathbf{x}, \mathbf{a}(t) = \mathbf{a}\right]. \quad (5)$$

The Q-function of the optimal policy, $Q_{\gamma}^{\pi^*}(\mathbf{x}, \mathbf{a})$, can be further written in a recursive form, known as the optimal Bellman equation

$$Q_{\gamma}^{\pi^{*}}\left(\mathbf{x}, \mathbf{a}\right) = \sum_{\mathbf{x}' \in \mathcal{X}} P_{\mathbf{x} \to \mathbf{x}'}^{\mathbf{a}} \left(R_{\mathbf{x} \to \mathbf{x}'}^{\mathbf{a}} + \gamma \max_{\mathbf{a}' \in \mathcal{A}} Q_{\gamma}^{\pi^{*}} \left(\mathbf{x}', \mathbf{a}' \right) \right),$$
(6)

where \mathbf{x}' denotes the next state $\mathbf{x}(t+1)$. Once $Q_{\gamma}^{\pi^*}(\mathbf{x}, \mathbf{a})$ is obtained, the optimal policy can be easily determined as

$$\pi^*(\mathbf{x}) = \arg\max_{\mathbf{a} \in \mathcal{A}} Q_{\gamma}^{\pi^*}(\mathbf{x}, \mathbf{a}). \tag{7}$$

In the environment with known transition tuple, the agent will learn the optimal policy not requiring real form of interactions with the environment. This can be achieved through solving the Bellman equation utilizing value iteration or policy iteration algorithm. However, it is difficult for the agent to know the tuple elements in reality, especially the transition probability and the reward function. To explore the unknown environment, Q-learning [46] is available to solve the Bellman equation and learn the statistics determined by the environment.

B. Q-learning Algorithm for Small-Scale Solution

In an unknown environment, the agent needs to continuously interact with the environment to improve the policy. Specifically, assuming the starting time slot for training is t, the agent should carry out the chosen action $\mathbf{a}(t)$ based on the current policy π_t and then observe the new state $\mathbf{x}(t+1)$ and the obtained reward $R_{\mathbf{x}(t) \to \mathbf{x}(t+1)}^{\mathbf{a}(t)}$. Thereafter, the agent updates

the Q-function from Q_t to Q_{t+1} , and accordingly improves the policy from π_t to π_{t+1} based on

$$\pi_{t+1}(\mathbf{x}(t)) = \arg\max_{\mathbf{a} \in A} Q_{t+1}(\mathbf{x}(t), \mathbf{a}). \tag{8}$$

When the action space is continuous, it is difficult to find the global maximum value for the non-convex function in (8). Hence each element in a is discretized into L levels uniformly over the interval [0,1]. To determine the valid actions in \mathcal{A} , we first select the action vectors satisfying $\sum_{i=1}^{C} a_i(t) = K$. To reduce the complexity, we further shrink the valid action space filtrated by the first constraint by using the prior knowledge of the cooperative coded caching. That is, only the action vectors satisfying $a_i(t) \leq \frac{l_0}{L}, \forall i = 1, \dots, C$, will be selected as the final action candidates, where $l_0 = \lceil \frac{L}{d} \rceil$ with $\lceil \cdot \rceil$ denoting the ceiling function. This is because taking the actions with $a_i(t) > \frac{l_0}{L}, \exists i = 1, \dots, C$, will waste the limited cache storage and thus corrupt the performance. Then the cardinality of the action space, denoted by $|\mathcal{A}|$, becomes limited. In addition, a finite number of possible popularity candidates, denoted by $|\Theta|$, is considered. Accordingly, the cardinality of the state space will be $|\mathcal{X}| = |\Theta||\mathcal{A}|$.

The state-action value function can be expressed by a table, i.e., Q-table, of which each element can be approximated by the average cumulative discounted rewards using the temporal difference learning and is give by

$$Q_{t+1}\left(\mathbf{x}(t), \mathbf{a}(t)\right) \leftarrow Q_{t}\left(\mathbf{x}(t), \mathbf{a}(t)\right) + \lambda \left[R_{\mathbf{x}(t) \to \mathbf{x}(t+1)}^{\mathbf{a}(t)} + \gamma \max_{\mathbf{a}' \in \mathcal{A}} Q_{t}\left(\mathbf{x}(t+1), \mathbf{a}'\right) - Q_{t}\left(\mathbf{x}(t), \mathbf{a}(t)\right)\right],$$
(9)

where λ is step size parameter.

According to the approximation method of the state-action value in (9) and the updating rule of the policy in (8), the Qlearning algorithm based cooperative coded caching strategy is described in Algorithm 1. The data used for learning the optimal policy are generated by the agent and an environment simulator. The content requests of users are generated based on a Zipf-like distribution [44] by the simulator. With the selected caching decision vector of SBSs, the simulator generates the next state and the reward. The initial values in the O-table are set as 0. The policy used for deciding the caching decision vector is first set randomly and then is improved with the updated Q-table. In the MDP case, it has been proved in [45] that all the state-action values in the Q-table will converge to the optimal values with probability 1 under the assumption that all the state-action values are updated for an infinite number of times and the stochastic approximation conditions on λ . Thus, the choice of $\mathbf{a}(t)$ under state $\mathbf{x}(t)$ in (9) follows ϵ -greedy approach to balance the exploitation and exploration and ensure the convergence. Then the optimal policy is accordingly obtained with the optimal state-action values.

In practice, the Q-learning algorithm based solution applies only to small-scale system, i.e., involving small numbers of SBSs and contents and a small cache size. That is because both $|\mathcal{A}|$ and $|\mathcal{X}|$ are related to the number of SBSs, p, the number of contents, C, the cache size, K, and the action discretization

Algorithm 1 Q-learning based Cooperative Coded Caching Strategy

Input: environment simulator, action space \mathcal{A} Output: the optimal caching policy $\pi^*(\mathbf{x})$ Procedure:

- 1: Set the starting time slot $t=t_0$, the starting state $\mathbf{x}(t)=\mathbf{x}_0$, the discount factor γ , the step size λ , the exploration probability ϵ , the initial value of Q-table: $Q_t(\mathbf{x},\mathbf{a})=0, \forall \mathbf{x},\mathbf{a}$, and for each state, select an arbitrary \mathbf{a} as its initial action
- 2: **loop**
- 3: (Selection of Cooperative Caching Action): Choose a caching action based on ε -greedy approach during offpeak hours

$$\mathbf{a}(t) = \pi_t^{\epsilon}(\mathbf{x}(t)) = \begin{cases} \pi_t(\mathbf{x}(t)), & \text{w.p. } 1 - \epsilon \\ \forall \mathbf{a} \in \mathcal{A}, & \text{w.p. } \epsilon \end{cases}$$

- 4: (MDS Coding): MBS codes the contents with $a_c(t) \neq 0$ and $a_c(t) \geq a_c(t-1)$ based on MDS coding during offpeak hours
- 5: (Coded Packets Delivery): MBS sends p different packets, each containing $(a_c(t) a_c(t-1))B$ bits of the cth coded content, to p SBSs correspondingly for caching during off-peak hours
- 6: (Cooperative Transmission): User $g \in \mathcal{G}_s$ selects d SBSs to form \mathcal{S}_g^d and requests contents during peak hours of time slot t+1
- 7: (New State Observation): MBS observes N(t + 1), computes $\theta(t + 1)$ according to (1) during off-peak hours, and sets the new state $\mathbf{x}(t+1) = [\theta(t+1), \mathbf{a}(t)]$
- (Reward Feedback): Environment feeds back a reward computed as (3)
- 9: (Estimation of Cooperative Caching Action): Estimate a caching action under the state $\mathbf{x}(t+1)$ according to

$$\tilde{\mathbf{a}}(t+1) = \arg \max_{\mathbf{a}' \in \mathcal{A}} Q_t \left(\mathbf{x}(t+1), \mathbf{a}' \right)$$

10: (*Q-table Updating*): Update the Q-table: for $\mathbf{x} = \mathbf{x}(t)$ and $\mathbf{a} = \mathbf{a}(t)$,

$$Q_{t+1}(\mathbf{x}, \mathbf{a}) = Q_t(\mathbf{x}, \mathbf{a}) + \lambda \left(R_{\mathbf{x}(t) \to \mathbf{x}(t+1)}^{\mathbf{a}(t)} + \gamma Q_t(\mathbf{x}(t+1), \tilde{\mathbf{a}}(t+1)) - Q_t(\mathbf{x}, \mathbf{a}) \right)$$

otherwise,

$$Q_{t+1}(\mathbf{x}, \mathbf{a}) = Q_t(\mathbf{x}, \mathbf{a})$$

11: (Policy Updating): Update the policy: for $\mathbf{x} = \mathbf{x}(t)$,

$$\pi_{t+1}(\mathbf{x}) = \arg\max_{\mathbf{a} \in A} Q_{t+1}(\mathbf{x}, \mathbf{a})$$

otherwise,

$$\pi_{t+1}(\mathbf{x}) = \pi_t(\mathbf{x})$$

- 12: t = t + 1
- 13: end loop
- 14: **return** The optimal caching policy $\pi^*(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}$

level, L, in our MDP case. In the large-scale system, large $|\mathcal{A}|$ and $|\mathcal{X}|$ make it impossible to save the Q-table with huge size. Meanwhile, it may be time-consuming to converge since many state-action pairs are seldom visited. Furthermore, the Q-learning algorithm performs ergodic search of the action space \mathcal{A} in steps 9 and 11 of each iteration, which leads to high time complexity. Therefore, we develop an efficient value function approximation based algorithm for the large-scale system in the next section.

IV. VALUE FUNCTION APPROXIMATION FOR LARGE-SCALE SOLUTION

In this section, we develop a value function approximation algorithm based solution for large-scale cooperative coded caching system. The state-action value function is first approximated, based on which the policy is updated without needing ergodic search in the action space. Meanwhile, the parameters in the approximate expression are updated using the stochastic gradient descent (SGD) method.

The basic idea of function approximation here is to denote the state-action value function $Q(\mathbf{x}, \mathbf{a})$ using a parameterized function approximator $\widehat{Q}(\mathbf{x}, \mathbf{a}), \forall \mathbf{x}, \mathbf{a}$, by taking into consideration the goal of the practical caching design. Since the state-action value function is a cumulation of the discounted rewards, inspired by (3), it can be approximated as

$$\widehat{Q}(\mathbf{x}(t), \mathbf{a}(t)) = \beta - \omega_1 \sum_{i=1}^{C} \eta_i \theta_i(t) (1 - da_i(t)) u (1 - da_i(t))$$

$$-\omega_2 \sum_{i=1}^{C} \xi_i (a_i(t) - a_i(t-1)) u (a_i(t) - a_i(t-1)),$$
(10)

where $u(\cdot)$ is the unit step function that is equal to 1 if the value in the parentheses is equal or greater than 0 and 0 otherwise, β , η_i , and ξ_i are unknown parameters, ω_1 and ω_2 represent weights of the corresponding parts, which are generally set as $\omega_1 \gg \omega_2$ because the first part refers to the reduplicative requests of users during peak hours while the second part represents the non-repetitive requests of the p SBSs during off-peak hours.

Instead of comparing the state-action values under all actions in each iteration in Algorithm 1, the caching action under the current state, $\mathbf{x}(t)$, can be obtained from the specific expression of the approximated value function and written as

$$\mathbf{a}(t) = \arg \max_{\mathbf{a}} \widehat{Q}(\mathbf{x}(t), \mathbf{a})$$

$$= \arg \max_{a_i, i \in \mathcal{C}} \left[\beta - \omega_1 \sum_{i=1}^{C} \eta_i \theta_i(t) (1 - da_i) u (1 - da_i) - \omega_2 \sum_{i=1}^{C} \xi_i (a_i - a_i(t-1)) u (a_i - a_i(t-1)) \right]. \quad (11)$$

Since ω_1 is much larger than ω_2 , the alternative action can be obtained by omitting $\omega_2 \sum_{i=1}^C \xi_i(a_i - a_i(t-1)) u(a_i - a_i(t-1))$

as

$$\hat{\mathbf{a}}(t) = \arg\max_{a_i, i \in \mathcal{C}} \left[\beta - \omega_1 \sum_{i=1}^C \eta_i \theta_i(t) (1 - da_i) u (1 - da_i) \right]. \tag{12}$$

The procedure of solving (12) is summarized as follows.

1) Determine the largest element of the action vector according to

$$l_{\max} = \lceil \frac{L}{d} \rceil,\tag{13}$$

where $l_{\max} \in \mathbb{N}^+$ denotes the numerator of the largest element.

2) Assume the number of contents corresponding to the caching fraction $\frac{i}{L}$ is z_i for $i=1,2,\ldots,l_{\max}$ and z_i is computed as

$$z_{i} = \begin{cases} \left\lfloor \frac{KL}{l_{\max}} \right\rfloor & i = l_{\max} \\ \left\lfloor \frac{KL - \sum_{j=1}^{l_{\max} - i} (i+j)z_{i+j}}{i} \right\rfloor & i = 1, 2, \dots, l_{\max} - 1. \end{cases}$$
(14)

3) Sort all the coefficients $\eta_i\theta_i(t)$, $i=1,2,\ldots,C$, as a vector in decending order and the jth element is $\eta_{h_j}\theta_{h_j}(t)$ corresponding to the h_j th content before sorting for $j=1,2,\ldots,C$. First, we roughly assign a value to each element of $\hat{\mathbf{a}}(t)$, which is given by

$$\hat{a}_{h_j}(t) = \begin{cases} \frac{l_{\max}}{L} & j = 1, 2, \dots, z_{l_{\max}} \\ \frac{l_{\max} - 1}{L} & j = z_{l_{\max}} + 1, z_{l_{\max}} + 2, \dots, z_{l_{\max}} + z_{l_{\max} - 1} \\ \vdots & \vdots & \vdots \\ \frac{1}{L} & j = \sum_{v = 0}^{l_{\max} - 2} z_{l_{\max} - v} + 1, \dots, \sum_{v = 0}^{l_{\max} - 2} z_{l_{\max} - v} + z_{1} \\ 0 & \text{others.} \end{cases}$$

Next, fine tune $\hat{a}_{h_j}(t)$ for $j=1,2,\ldots,z_{l_{\max}}$ with $1-d\cdot \frac{l_{\max}}{L}<0$. Starting with $j=z_{l_{\max}}$, repeat the following steps until j=1. For $j^{'}=z_{l_{\max}}+z_{l_{\max}-1}+1,\ldots,C$, find the minimum value of $j^{'}$, i.e. $j^{'}$, satisfying

$$\left(1 - d \cdot \frac{l_{\max} - 1}{L}\right) \cdot \eta_{h_j} \theta_{h_j}(t) < d \cdot \frac{1}{L} \cdot \eta_{h_{\hat{j}'}} \theta_{h_{\hat{j}'}}(t).$$
(16)

Then adjust $\hat{a}_{h_j}(t)$ by subtracting $\frac{1}{L}$ and $\hat{a}_{h_{\hat{j}'}}(t)$ by adding $\frac{1}{L}$.

The objective is to make the parameterized state-action value function $\widehat{Q}(\mathbf{x}, \mathbf{a})$ and the real $Q(\mathbf{x}, \mathbf{a})$ as close as possible. The loss function is defined as

$$Loss(\beta, \eta, \xi) = \mathbb{E}\left[\left(Q(\mathbf{x}, \mathbf{a}) - \widehat{Q}(\mathbf{x}, \mathbf{a})\right)^{2}\right], \quad (17)$$

where $\eta = [\eta_1, \eta_2, \dots, \eta_C]$ and $\boldsymbol{\xi} = [\xi_1, \xi_2, \dots, \xi_C]$. $Q(\mathbf{x}, \mathbf{a})$ is unknown and can be replaced by the currently estimated value function, $R^{\mathbf{a}}_{\mathbf{x} \to \mathbf{x}'} + \gamma \widehat{Q}\left(\mathbf{x}', \widetilde{\mathbf{a}}\right)$, resorting to the temporal difference learning. The estimated caching action under state \mathbf{x}' , which is denoted by $\widetilde{\mathbf{a}} = \arg\max_{\mathbf{a}'} \widehat{Q}\left(\mathbf{x}', \mathbf{a}'\right)$, can also be obtained following the same procedure as the above. Based on each sample $(\mathbf{x}(t), \mathbf{a}(t))$, the parameters, $\{\beta, \eta, \xi\}$, are

updated resorting to the SGD method to minimize the loss function as

$$\beta^{\text{cu}} = \beta^{\text{pr}} - \delta \frac{\partial Loss}{\partial \beta}$$

$$= \beta^{\text{pr}} + 2\delta \left(Q(\mathbf{x}(t), \mathbf{a}(t)) - \widehat{Q}(\mathbf{x}(t), \mathbf{a}(t)) \right), \quad (18a)$$

$$\eta_{i}^{\text{cu}} = \eta_{i}^{\text{pr}} - \delta \frac{\partial Loss}{\partial \eta_{i}}$$

$$= \eta_{i}^{\text{pr}} - 2\delta\omega_{1}\theta_{i}(t) \left(Q(\mathbf{x}(t), \mathbf{a}(t)) - \widehat{Q}(\mathbf{x}(t), \mathbf{a}(t)) \right)$$

$$\times (1 - da_{i}(t))u \left(1 - da_{i}(t) \right), \quad (18b)$$

$$\xi_{i}^{\text{cu}} = \xi_{i}^{\text{pr}} - \delta \frac{\partial Loss}{\partial \xi_{i}}$$

$$= \xi_{i}^{\text{pr}} - 2\delta\omega_{2} \left(Q(\mathbf{x}(t), \mathbf{a}(t)) - \widehat{Q}(\mathbf{x}(t), \mathbf{a}(t)) \right)$$

$$\times (a_{i}(t) - a_{i}(t - 1))u \left(a_{i}(t) - a_{i}(t - 1) \right), \quad (18c)$$

where $\beta^{\rm cu}$, $\eta_i^{\rm cu}$, and $\xi_i^{\rm cu}$ denote the parameters in the current time slot, $\beta^{\rm pr}$, $\eta_i^{\rm pr}$, and $\xi_i^{\rm pr}$ represent the parameters in the previous time slot, and δ is the step size.

Based on the procedure to find the caching action and the method to update the parameters described above, the value function approximation based cooperative coded caching strategy is summarized in Algorithm 2 in detail.

V. SIMULATION RESULTS

In this section, simulation results are presented to verify the proposed RL based cooperative coded caching strategies for small-scale and large-scale systems. Unless stated otherwise, the Zipf-like distribution is used to generate the user requests, based on which the content popularity profiles can be computed according to (1). The following four baselines are used for performance comparison.

- Baseline 1 (Value Iteration based Optimal Caching): The state value function, V_γ^π(x), denotes the expected cumulative discounted rewards following the policy π under the current state x. The state value function of the optimal policy is expressed as V_γ^{π*}(x) = max Q_γ^{π*}(x, a). Under the assumption that the agent knows the transition tuple described in III-A, the state value function for each x is updated according to V(x) = max D_{x'∈X} P^a_{x→x'} × (R^a_{x→x'} + γV(x')) until converging to V_γ^{π*}(x). Then the optimal policy is obtained as π*(x) = arg max Q_γ^{π*}(x, a).
 Baseline 2 (Most Popular based Cooperative Caching
- **Baseline 2** (Most Popular based Cooperative Caching (MPCC)): According to $\theta(t)$ at time slot t, the agent selects the most popular contents to cache incorporating the MDS coding while the interactions with the environment are not considered, i.e., without RL. It can be seen how this baseline scheme caches contents based on $\theta(t)$ from Fig. 8.
- Baseline 3 (RL based Non-cooperative Caching (RL-NC)): The agent determines the caching strategy based on RL while the cooperation among SBSs is not considered, which corresponds to a special case when d = 1 in our proposed algorithms. In this case, full contents are cached at the SBSs.

Algorithm 2 Value Function Approximation based Cooperative Coded Caching Strategy

Input: environment simulator, the structure of the value function approximator $Q(\mathbf{x}, \mathbf{a})$

Output: the value function approximator $\hat{Q}(\mathbf{x}, \mathbf{a})$

Procedure:

1: Set the starting time slot $t = t_0$, the starting state $\mathbf{x}(t) =$ \mathbf{x}_0 , the discount factor γ , the step size δ , the exploration probability ϵ , the initial value of parameters: $\beta^{pr} = 0$, $\eta^{\mathrm{pr}} = 0, \, \boldsymbol{\xi}^{\mathrm{pr}} = 0$

2: **loop**

(Selection of Cooperative Caching Action): Choose a caching action based on ε -greedy approach during offpeak hours

w.p. $1 - \varepsilon$

solve $\mathbf{a}(t) = \arg\max \hat{Q}\left(\mathbf{x}(t), \mathbf{a}\right)$ based on (12) and the procedure 1), 2), 3

choose an action randomly that meets the constraints $\sum_{i=1}^{C} a_i(t) = K$ and $l_{\max} = \lceil \frac{L}{d} \rceil$

- (MDS Coding): MBS codes the contents with $a_c(t) \neq 0$ and $a_c(t) \ge a_c(t-1)$ based on MDS coding during offpeak hours
- (Coded Packets Delivery): MBS sends p different packets, each containing $(a_c(t) - a_c(t-1))B$ bits of the cth coded content, to p SBSs correspondingly for caching during off-peak hours
- (Cooperative Transmission): User $g \in \mathcal{G}_s$ selects dSBSs to form \mathcal{S}_q^d and requests contents during peak hours of time slot t+1
- (New State Observation): MBS observes N(t + 1), 7: computes $\theta(t+1)$ according to (1) during off-peak hours, and sets the new state $\mathbf{x}(t+1) = [\boldsymbol{\theta}(t+1), \mathbf{a}(t)]$
- (Reward Feedback): Environment feeds back a reward 8: computed as (3)
- (Estimation of Cooperative Caching Action): Estimate a caching action under the state x(t + 1) by solving $\tilde{\mathbf{a}}(t+1) = \arg\max_{\mathbf{a}'} \hat{Q}\left(\mathbf{x}(t+1), \mathbf{a}'\right)$ based on (12) and the procedure 1), 2), 3)
- (Parameters Updating): Update the 10: parameters $\{\beta, \eta, \xi\}$ based on (18)
- t = t + 111:
- 12: end loop
- 13: **return** The value function approximator $\hat{Q}(\mathbf{x}, \mathbf{a})$
 - Baseline 4 (RL based Uncoded Cooperative Caching (RL-UCC)): The agent determines the caching strategy based on RL with SBS cooperation while the fragments stored at the SBSs are uncoded. Specifically, each SBS randomly stores the fragments with the corresponding size according to the action made by the agent.

By comparing the proposed algorithms with Baselines 2, 3, and 4, respectively, the effects of RL, the combination of MDS coding and SBS cooperation, and MDS coding employed in the proposed algorithms will be clear.

TABLE I SIMULATION PARAMETERS IN SMALL-SCALE SYSTEM

Simulation Parameter	Setting Value
\overline{p}	20
\overline{C}	10
K	[1,2,3,4]
d	2
L	3
γ	0.9
λ	0.6
α_1	1.36
α_2	2.3
ϵ for exploration stage	0.1
ϵ for exploitation stage	0

To measure and compare various schemes, define the direct SBS-serving ratio as

$$\rho = \frac{N_{\rm SBS}}{N_{\rm total}},\tag{19}$$

where $N_{\rm SBS}$ is the traffic load served directly by the SBSs and N_{total} represents the total traffic load.⁶

A. Performance in Small-Scale System

In the small-scale system, two content popularity profiles, θ_1 and θ_2 , are considered and computed according to (1), where the user requests are generated based on the Zipf-like distribution [44] with α_1 and α_2 denoting the corresponding skewness. Using the case with the starting state $[\theta_1, \mathbf{a}_i]$ and the selected action a_i as an example, the transition probabilities are chosen randomly from [0, 1] and subject to the following constraints

$$\begin{split} P_{[\boldsymbol{\theta}_{1},\mathbf{a}_{i}]\rightarrow[\boldsymbol{\theta}_{1},\mathbf{a}_{k}]}^{\mathbf{a}_{j}} + P_{[\boldsymbol{\theta}_{1},\mathbf{a}_{i}]\rightarrow[\boldsymbol{\theta}_{2},\mathbf{a}_{k}]}^{\mathbf{a}_{j}} &= 1 \quad k = j, \qquad \text{(20a)} \\ P_{[\boldsymbol{\theta}_{1},\mathbf{a}_{i}]\rightarrow[\boldsymbol{\theta}_{1},\mathbf{a}_{k}]}^{\mathbf{a}_{j}} &= 0 \quad k \neq j, \qquad \text{(20b)} \end{split}$$

$$P_{[\boldsymbol{\theta}_1, \mathbf{a}_k] \to [\boldsymbol{\theta}_1, \mathbf{a}_k]}^{\mathbf{a}_j} = 0 \quad k \neq j, \tag{20b}$$

$$P_{[\boldsymbol{\theta}_1, \mathbf{a}_i] \to [\boldsymbol{\theta}_2, \mathbf{a}_k]}^{\mathbf{a}_j} = 0 \quad k \neq j.$$
 (20c)

Major simulation parameters in the small-scale system are listed in Table I.

To compare caching efficiency, Fig. 5 plots the direct SBSserving ratio, ρ , against growing cache size, K, for Algorithm 1, the MPCC scheme, the RL-NC scheme, and the RL-UCC scheme in the small-scale system. From the figure, the performance improves with cache size for all caching schemes and the proposed Algorithm 1 performs better than the other three baselines. By comparing Algorithm 1 and MPCC, we can see the significant advantage of using RL to make caching decision. Algorithm 1 also outperforms RL-NC resorting to the combination of MDS coding and SBS cooperation. The RL-UCC scheme achieves better performance than MPCC and RL-NC but still has a significant gap to Algorithm 1 since the MDS coding based caching considered in Algorithm 1 can utilize the limited cache storage more efficiently than the uncoded random caching used in RL-UCC. In more details, each SBS caches the individual coded packet irrelevant to each other in the MDS coding based caching, which guarantees

 6 In fact, $N_{
m SBS}$ is also the reward in our formulated RL problem and $N_{
m total}$ is also the total number of content requests.

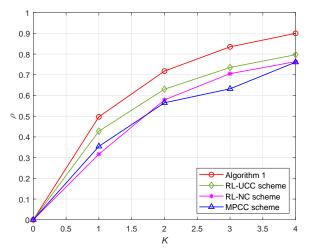


Fig. 5. The direct SBS-serving ratio for Algorithm 1, the MPCC scheme, the RL-NC scheme, and the RL-UCC scheme in the small-scale system.

the successful content recovery so long as the user collects a certain number of packets with the total size no less than the size of the original content from the SBSs. In the uncoded random caching, the SBSs cache the uncoded fragments of each content randomly and thus the user may receive repetitive fragments from the SBSs, in which case the performance of RL-UCC is degraded. For example, Algorithm 1 achieves performance gains of about 0.14, 0.14, and 0.1 compared to MPCC, RL-NC, and RL-UCC, respectively, when the cache size K = 4. We can also find that the effect of RL is larger than the MDS coding in the small-scale system by comparing MPCC and RL-UCC. To converge to the corresponding ρ , Algorithm 1, RL-UCC, and RL-NC consume 375.8 seconds, 372.3 seconds, and 80.5 seconds, respectively, when K=1. The corresponding time increases to 1034.6 seconds, 1300.9 seconds, and 110.2 seconds, respectively, when K=2. So the consumed time for convergence is dependent on the size of the action space. In addition, SBS cooperation will enlarge the action space, which is the reason that Algorithm 1 and RL-UCC use more time than RL-NC.

To show the convergence behaviors of the proposed Algorithm 1 and the baseline schemes, Fig. 6 plots the direct SBS-serving ratio, ρ , with increasing time slot, t, with the cache size K=1 and 2. From Fig. 6(a), the performance of MPCC stabilizes at about 0.35 fast. In contrast, Algorithm 1 achieves the increasing performance with t in the exploration stage with $t<10^5$ and finally converges to the optimal value, $\rho=0.5$, by setting $\epsilon=0$ in the exploitation stage starting from $t=10^5$, demonstrating the effectiveness of caching policy learnt through interactions with the environment. The performance of RL-UCC has the similar trend but suffers a degradation due to using the less efficient uncoded random caching. More performance loss is incurred for RL-NC since both SBS cooperation and MDS coding are not utilized. The action space enlarges with K=2, based on which we allocate

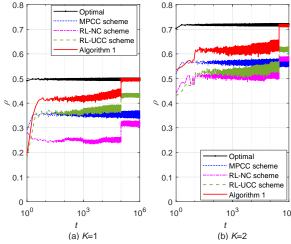


Fig. 6. The convergence performance of Algorithm 1 and baseline schemes with K=1 and 2.

more time slots for the exploration stage and set $\epsilon=0$ at $t=3\times 10^5$ in Fig. 6(b). Compared to Fig. 6(a), Fig. 6(b) shows the similar convergence behavior and increased converged performance due to the larger cache size for each scheme.

To provide an intuitive explanation about the performance gain achieved by Algorithm 1, we plot the curve of cosine similarity with K=1 and 2 in Fig. 7. The cosine similarity is defined as $\cos(\phi_{\mathbf{a}_{\mathrm{opt}},\Delta}) = \frac{\langle \mathbf{a}_{\mathrm{opt}}^T,\Delta^T \rangle}{\|\mathbf{a}_{\mathrm{opt}}\|\cdot\|\Delta\|}$ with $\langle \cdot, \cdot \rangle, \|\cdot\|$, and $(\cdot)^T$ representing the inner product, Euclidean norm, and transpose, where $\mathbf{a}_{\text{opt}}, \Delta$, and $\phi_{\mathbf{a}_{\text{opt}},\Delta}$ denote the optimal action vector, the corresponding action vector of Algorithm 1, MPCC, RL-NC, and RL-UCC, and the angle between \mathbf{a}_{opt} and Δ , respectively. From Fig. 7(a), the similarity curves of Algorithm 1 and the other three baseline schemes are on the rise as t increases until reaching their respective convergence values. The curves of MPCC and RL-NC stabilize at about 0.8 and 0.6, respectively, while the curves of Algorithm 1 and RL-UCC finally converge to 1. That is, the superiority of Algorithm 1 to MPCC and RL-NC mainly owes to the effect of RL and the effect of the combination of SBS cooperation and MDS coding, respectively. Both Algorithm 1 and RL-UCC are able to learn the optimal action, which demonstrates that Algorithm 1 outperforms RL-UCC resorting to the advantage of MDS coding based caching over the uncoded random caching. For Fig. 7(b) with K=2, Algorithm 1 and RL-UCC can still find the optimal action while the similarity performance of RL-NC and MPCC finally converges to 0.82 and 0.8, respectively. Through Fig. 7, the effects of RL, SBS cooperation, and MDS coding are revealed clearly.

To get more insights, we use a toy example to show how they take action to match the content popularity in Fig. 8. Specifically, Fig. 8(a) plots the popularity profiles of three successive time slots and Fig. 8(b) plots the action of each scheme at the end of the first time slot 110010. From the figure, the MPCC scheme simply selects the action based on the popularity profile of time slot 110010, which obviously mismatches the upcoming popularity profile. RL-NC, RL-UCC, and Algorithm 1 all select action based on the long-term performance. However, RL-NC only caches full content and

 $^{^{7}}$ It is noted that the switch point, $t=10^{5}$, from the exploration stage to the exploitation stage is determined according to the simulation trials. Setting ϵ to 0 untimely will lead to insufficient exploration in action space and consequently degenerate the converged performance.

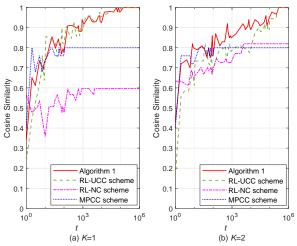
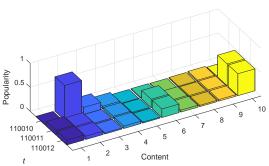
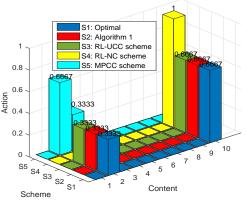


Fig. 7. Cosine similarity between the optimal action vector and those of Algorithm 1, the MPCC scheme, the RL-NC scheme, and the RL-UCC scheme with K=1 and 2.



(a) Popularity profiles of 3 successive time slots.



(b) Actions of different schemes at the end of the first time slot.

Fig. 8. A toy example: Popularity profiles and corresponding actions with $K=1.\,$

inevitably causes performance loss. In contrast, RL-UCC and Algorithm 1 smartly allocate the action to the most popular contents in all popularity profiles, which is the optimal action. This again reveals that the uncoded random caching used by RL-UCC wastes the limited cache storage and thus performs worse than Algorithm 1 even though they have learned very similar caching policy.

TABLE II SIMULATION PARAMETERS IN LARGE-SCALE SYSTEM

Simulation Parameter	Setting Value
\overline{p}	50
\overline{C}	100
K	[5,10,15,20]
\overline{d}	3
L	6
${\gamma}$	0.9
δ for Algorithm 2	[0.01,0.01,0.1,0.5]
and RL-UCC scheme	corresponds to K
δ for RL-NC scheme	[0.01,0.01,0.8,0.8]
	corresponds to K
ϵ for exploration stage	0.2
ϵ for exploitation stage	0

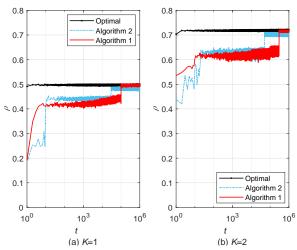


Fig. 9. The convergence performance of Algorithm 2 with K=1 and 2 in the small-scale system.

B. Performance in Large-Scale System

In the large-scale system, ten content popularity profiles are considered and computed according to (1), where the user requests are generated based on Zipf-like distribution [44] with $\alpha_i = 1.2 + 0.2(i-1)$, for $i=1,\ldots,10$, denoting the skewness of the ith user request distribution. And the transition probabilities are generated randomly from [0,1] under the constraints $\sum_{j=1}^{10} P_{[\theta_i,\mathbf{a}] \to [\theta_j,\mathbf{a}']}^{\mathbf{a}'} = 1$, for $i=1,\ldots,10$, where \mathbf{a}' is the chosen action under the state $[\theta_i,\mathbf{a}]$. Major simulation parameters in the large-scale system are listed in Table II.

Fig. 9 shows the convergence performance of Algorithm 2, where the optimal scheme and Algorithm 1 are also plotted to demonstrate the near-optimality and fast convergence of Algorithm 2. All the schemes in this figure still use the parameters listed in Table I since the optimal scheme and Algorithm 1 can be obtained only in the small-scale system due to high computational complexity. Although Algorithm 2 is designed for the large-scale system, it is also feasible in the small-scale system. In Fig. 9(a) and (b), Algorithm 2 employs value function approximation to avoid the infinite execution for each state-action pair and the onerous ergodic search over the action space. As a result, Algorithm 2 converges faster than Algorithm 1 at the cost of only converging to the

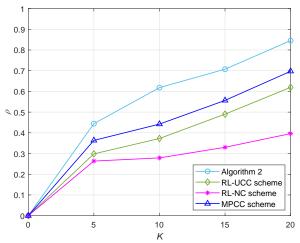


Fig. 10. The direct SBS-serving ratio for Algorithm 2, the MPCC scheme, the RL-NC scheme, and the RL-UCC scheme in the large-scale system.

near-optimal point. Nevertheless, Algorithm 2 can be flexibly applied to the large-scale system, in which case Algorithm 1 is infeasible. From the figure, the exploitation-only performance of Algorithm 1 is better than that of Algorithm 2 during the exploitation stage ($\epsilon=0$), which is because it finds the optimal policy through exhaustive search while Algorithm 2 only converges to a sub-optimal solution with the approximated value function. However, during the exploration stage ($\epsilon>0$), the exploration in the action space brings some uncertainties to the exploration-exploitation performance. As a result, Algorithm 1 may achieve the performance similar to Algorithm 2 or outperform Algorithm 2 at the switch point.

Fig. 10 plots the direct SBS-serving ratio, ρ , versus the cache size, K, for Algorithm 2, the MPCC scheme, the RL-NC scheme, and the RL-UCC scheme in the large-scale system, where Algorithm 2 outperforms the other three baselines more significantly than in the small-scale system. When the cache size K = 15, Algorithm 2 achieves the direct SBS-serving ratio $\rho = 0.71$ while the corresponding performance of MPCC, RL-NC, and RL-UCC are 0.55, 0.33, and 0.49 respectively, which means that Algorithm 2 can achieve the considerable performance even with limited cache size. The effectiveness of Algorithm 2 remedies the infeasibility of Algorithm 1 in the large-scale system and makes our proposed RL based cooperative coded caching scheme applicable to various practical systems. In addition, the superiority of MPCC over RL-UCC demonstrates that the uncoded random caching causes more significant performance loss and the MDS coding based caching becomes more important in the large-scale system.

To verify the robustness of Algorithm 2, we introduce the temporal locality to the user requests based on the shot noise model (SNM) according to [47], where a small amount of contents are popular temporarily. Fig. 11 plots the direct SBS-serving ratio for Algorithm 2 and the three baseline schemes in the large-scale system under the SNM based user requests. Similar to Fig. 10, Algorithm 2 still achieves remarkable advantage compared to the baseline schemes. The SNM based user requests with temporal locality only cause slight performance loss for Algorithm 2.

To further demonstrate the robustness of Algorithm 2 in the

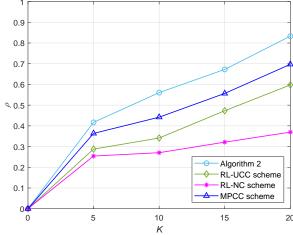


Fig. 11. The direct SBS-serving ratio for Algorithm 2, the MPCC scheme, the RL-NC scheme, and the RL-UCC scheme in the large-scale system under SNM based user requests.

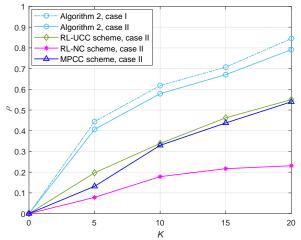


Fig. 12. The robustness of Algorithms 2 in the large-scale system.

TABLE III		
COMPLEXITY COMPARISON		

Scheme	Small-scale system	Large-scale system
MPCC		
RL-NC	$\mathcal{O}(T_{\mathrm{RN}} \mathcal{A}_{\mathrm{RN}})$	$\mathcal{O}(T_{\mathrm{RN}}C)$
RL-UCC	$\mathcal{O}(T_{\mathrm{RU}} \mathcal{A}_{\mathrm{RU}})$	$\mathcal{O}(T_{\mathrm{RU}}C)$
Algorithm 1	$ \begin{array}{c c} \mathcal{O}(T_1(\mathcal{A}_1 \\ +CB_c^{\text{MDS}} \log B_c^{\text{MDS}})) \end{array} $	-
Algorithm 2	$\mathcal{O}(T_2CB_c^{\text{MDS}}\log B_c^{\text{MDS}})$	$\mathcal{O}(T_2CB_c^{\text{MDS}}\log B_c^{\text{MDS}})$

time-varying environment, Fig. 12 plots the performance of Algorithm 2 and the baseline schemes in two cases, where case I uses the environment set in the first paragraph of Section V.B while case II uses another set of content popularity candidates and transition probabilities. From Fig. 12, the environment changing only leads to a minor performance loss for Algorithm 2. When facing case II with the changed environment, Algorithm 2 achieves the more significant advantage over other baseline schemes compared to Fig. 10.

C. Complexity Comparison

Table III lists the computational complexity of MPCC, RL-NC, RL-UCC, Algorithm 1, and Algorithm 2 when achieving the converged performance in small- and large-scale systems. $T_{\rm RN}$, $T_{\rm RU}$, $T_{\rm 1}$, and $A_{\rm RN}$, $A_{\rm RU}$, $A_{\rm 1}$ denote the total steps for convergence and action space of RL-NC, RL-UCC, and Algorithm 1, respectively. T_2 denotes the total steps for convergence of Algorithm 2. MPCC makes decision simply based on the current popularity profile without needing iteration and thus its complexity does not include the number of steps for convergence. RL-NC, RL-UCC, and Algorithm 1 in the smallscale system apply Q-learning with Q-table updating and the complexity, dependent on the size of the action space, will be prohibitively huge in the large-scale system. It can be seen that Algorithm 2 significantly reduces the complexity compared to Algorithm 1 and causes the complexity of only $\mathcal{O}(C)$ for parameter updating in each step. RL-NC and RL-UCC also use value function approximation similar to Algorithm 2 to reduce the complexity in the large-scale system. By using the comparable or moderately increased complexity, Algorithms 1 and 2 achieve the better performance than other baseline schemes.

For Algorithm 2, T_2 increases with the size of action space that depends on the number of contents, C, the action discretization level, L, and the cache size, K, while $B_c^{\rm MDS}$ increases with the number of SBSs, p, according to (2). The complexity reveals the key system parameters impacting the capability of Algorithm 2. Although the complexity of Algorithm 2 increases with p, C, and L, it is still feasible so long as these parameters are not too large. When they grow to very large values, we can omit the contents that are barely requested, which account for the majority of all contents [47]. We can also reduce L appropriately. With these solutions, Algorithm 2 is able to work in the very large-scale system at the cost of the acceptable performance loss.

D. Impact of Action Discretization

To reveal the impact of action discretization used in the proposed algorithms, we compare the deep deterministic policy gradient (DDPG) algorithm [32] that is able to output the continuous action with Algorithm 1 in the small-scale system and with Algorithm 2 in the large-scale system, respectively, in Fig. 13. From Fig. 13(a), Algorithm 1 achieves very similar performance to DDPG, indicating that the action discretization only causes a very limited loss.8 To achieve their respective performance at K=1, Algorithm 1 consumes only 375.8 seconds while DDPG needs 1296.6 seconds. DDPG needs much more time to find the appropriate continuous action, which however only brings a marginal performance improvement. For the large-scale case in Fig. 13(b), Algorithm 2 slightly outperforms DDPG even if it discretizes the action space. This reveals that finding the appropriate action from a continuous space becomes more difficult for DDPG since it has the complicated network structures of actor and critic and needs to update much more parameters in each iteration.

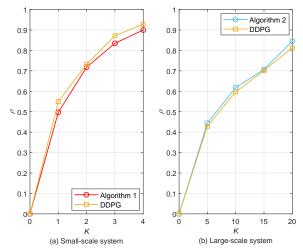


Fig. 13. Performance comparison between the proposed algorithms and DDPG.

On the other hand, Algorithm 2 needs 565.3 seconds to achieve its performance at K=5 while the time consumed by DDPG is 3411.9 seconds. Algorithm 2 benefits from the insightful approximation for the state-action value function and the efficient action selection in each iteration. Therefore, action discretization is preferable in our considered caching design problem since it hardly impacts the performance but reduces the computational complexity significantly.

VI. CONCLUSION

In this paper, we have studied the RL based cooperative coded caching strategy design for SBSs in UDNs. The practical time-variant popularity profile with the unknown evolution way is taken into consideration to formulate the MDS coding based cooperative caching design in the perspective of RL. The optimal solution is first developed by embedding the complicated cooperative MDS coding into Q-learning. For the large-scale system, we further propose an insightful approximation for the state-action value function heuristically along with an efficient action-selection approach, which is proved near-optimal but with significantly reduced complexity. The proposed RL based cooperative coded caching strategy has significant performance superiority and is applicable to practical systems with different dimensionalities and environments.

In the future, we can further consider that the SBSs use their respective caching policies in the multi-agent RL manner. For multi-agent RL, the agents impact each other in decision making and thus independent Q-learning based on own action and observation will lead to the nonstationary environment for each agent [17], which needs to be addressed in the distributed framework for caching policy design.

REFERENCES

- [1] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What will 5G be?" *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, Jun. 2014.
- [2] J. Hoadley and P. Maveddat, "Enabling small cell deployment with HetNet," *IEEE Wireless Commun.*, vol. 19, no. 2, pp. 4–5, Apr. 2012.
- [3] H. S. Dhillon, R. K. Ganti, F. Baccelli, and J. G. Andrews, "Modeling and analysis of K-tier downlink heterogeneous cellular networks," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 3, pp. 550–560, Apr. 2012.

⁸Algorithm 1 is the optimal solution for the discrete action case and thus has a tiny performance gap to DDPG.

- [4] ITU-R report M.2320: Future technology trends of terrestrial IMT systems, Nov. 2014. [Online]. Available: http://www.itu.int/dms_pub/ itu-r/opb/rep/R-REP-M.2320-2014-PDF-E.pdf.
- [5] Small Cell Forum, "Backhaul technologies for small cells," White Paper, 2013.
- [6] X. Ge, H. Cheng, M. Guizani, and T. Han, "5G wireless backhaul networks: challenges and research advances," *IEEE Network*, vol. 28, no. 6, pp. 6–11, Nov. 2014.
- [7] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. Leung, "Cache in the air: exploiting content caching and delivery techniques for 5G systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 131–139, Feb. 2014.
- [8] A. Liu and V. Lau, "Explicating base station caching in MIMO cellular networks: opportunistic cooperation for vedio streaming," *IEEE Trans. Signal Process.*, vol. 63, no. 1, pp. 57–69, Jan. 2015.
- [9] Intel, "Rethinking the small cell business model," White Paper, 2011.
- [10] F. Tang, Y. Kawamoto, N. Kato, and J. Liu, "Future intelligent and secure vehicular network towards 6G: Machine-learning approaches," *Proc. IEEE*, vol. 108, no. 2, pp. 292–307, Feb. 2020.
- [11] H. Huang, S. Guo, G. Gui, Z. Yang, J. Zhang, H. Sari, and F. Adachi, "Deep learning for physical-layer 5G wireless techniques: Opportunities, challenges and solutions," *IEEE Wirel. Commun. Mag.*, to be published.
- [12] Y. Wang, J. Yang, M. Liu, and G. Gui, "LightAMC: Lightweight automatic modulation classification using deep learning and compressive sensing," *IEEE Trans. Veh. Technol.*, to be published.
- [13] P. Dong, H. Zhang, G. Y. Li, I. Gaspar, and N. NaderiAlizadeh, "Deep CNN-based channel estimation for mmWave massive MIMO systems," *IEEE J. Sel. Topics Signal Process.*, vol. 13, no. 5, pp. 989–1000, Sep. 2019.
- [14] S. Gao, P. Dong, Z. Pan, and G. Y. Li, "Deep learning based channel estimation for massive MIMO with mixed-resolution ADCs," *IEEE Commun. Lett.*, vol. 23, no. 11, pp. 1989–1993, Nov. 2019.
- [15] L. Liang, H. Ye, G. Yu, and G. Y. Li, "Deep learning based wireless resource allocation with application to vehicular networks," *Proc. IEEE*, vol. 108, no. 2, pp. 341–356, Feb. 2020.
- [16] H. Ye, G. Y. Li, and B.-H. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3163–3173, Apr. 2019.
- [17] L. Liang, H. Ye, and G. Y. Li, "Spectrum sharing in vehicular networks based on multi-agent reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2282–2292, Oct. 2019.
- [18] K. Poularakis, G. Iosifidis, and L. Tassiulas, "Approximation algorithms for mobile data caching in small cell networks," *IEEE Trans. Commun.*, vol. 62, no. 10, pp. 3665–3677, Oct. 2014.
- [19] K. Poularakis, G. Iosifidis, V. Sourlas, and L. Tassiulas, "Exploiting caching and multicast for 5G wireless networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 4, pp. 2995–3007, Apr. 2016.
- [20] Y. Zhou, Z. Zhao, R. Li, H. Zhang, and Y. Louet, "Cooperation-based probabilistic caching strategy in clutered cellular networks," *IEEE Commun. Lett.*, vol. 21, no. 9, pp. 2029–2032, Sep. 2017.
- [21] Y. Cui and D. Jiang, "Analysis and optimization of caching and multicasting in large-scale cache-enabled heterogeneous wireless networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 1, pp. 250–264, Jan. 2017.
- [22] E. Baştuğ, M. Bennis, M. Kountouris, and M. Debbah, "Edge caching for coverage and capacity-aided heterogeneous networks," in *Proc. IEEE Int. Symp. Inf. Theory. (ISIT)*, Jul. 2016, pp. 285–289.
- [23] J. Wen, K. Huang, S. Yang, and V. O. K. Li, "Cache-enabled heterogeneous cellular networks: optimal tier-level content placement," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5939–5952, Sep. 2017.
- [24] C. Yang, Y. Yao, Z. Chen, and B. Xia, "Analysis on cache-enabled wireless heterogeneous networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 1, pp. 131–145, Jan. 2016.
- [25] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.
- [26] F. Gabry, V. Bioglio, and I. Land, "On energy-efficient edge caching in heterogeneous networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3288–3298, Dec. 2016.
- [27] Z. Chen, J. Lee, T. Q. S. Quek, and M. Kountouris, "Cooperative caching and transmission design in cluster-centric small cell networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 5, pp. 3401–3415, May 2017.
- [28] S. O. Somuyiwa, A. György, and D. Gündüz, "A reinforecement-learning approach to proactive caching in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1331–1344, Jun. 2018.
- [29] E. Rezaei, H. E. Manoochehri, and B. H. Khalaj, "Multi-agent learning for cooperative large-scale caching networks," arXiv preprint arXiv:1807.00207, Jun. 2018.

- [30] N. Zhang, K. Zheng, and M. Tao, "Using grouped linear prediction and accelerated reinforcement learning for online content caching," in *Proc. IEEE Int. Conf. Commun. Workshops*, May 2018, pp. 1–6.
- [31] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal and scalable caching for 5G using reinforcement learning of space-time popularities," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 180–190, Feb. 2018.
- [32] Y. Wei, F. R. Yu, M. Song, and Z. Han, "Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor-critic deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2061–2073, Apr. 2019.
- [33] Q. Ye, B. Rong, Y. Chen, M. Shalash, C. Caramanis, and J. G. Andrews, "User association for load balancing in heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 6, pp. 2706–2716, Jun. 2013.
- [34] H. S. Dhillon and G. Caire, "Wireless backhaul networks: capacity bound, scalability analysis and design guidelines," *IEEE Trans. Wireless Commun.*, vol. 14, no. 11, pp. 6043–6056, Nov. 2015.
- [35] H. H. Yang, G. Geraci, and T. Q. S. Quek, "Energy-efficient design of MIMO heterogeneous networks with wireless backhaul," *IEEE Trans. Wireless Commun.*, vol. 15, no. 7, pp. 4914–4927, Jul. 2016.
- [36] Nokia, "Ultra dense network (UDN)," White Paper, 2016.
- [37] X. Ge, S. Tu, G. Mao, C.-X. Wang, and T. Han, "5G ultra-dense cellular networks," *IEEE Wireless Commun.*, vol. 23, no. 1, pp. 72–79, Feb. 2016.
- [38] M. Kamel, W. Hamouda, and A. Youssef, "Ultra-dense networks: a survey," *IEEE Commun. Surv. Tut.*, vol. 18, no. 4, pp. 2522–2545, May 2016.
- [39] H. Jiang, Z. Pan, N. Liu, X. You, and T. Deng, "Gibbs-sampling-based CRE bias optimization algorithm for ultradense networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 1334–1350, Apr. 2016.
- [40] J. Du, E. Gelenbe, C. Jiang, H. Zhang, and Y. Ren, "Contract design for traffic offloading and resource allocation in heterogeneous ultra-dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2457–2467, Nov. 2017.
- [41] W. Wen, Y. Cui, F. Zheng, S. Jin, and Y. Jiang, "Random caching based cooperative transmission in heterogeneous wireless networks," *IEEE Trans. Commun.*, vol. 66, no. 7, pp. 2809–2825, Jul. 2018.
- [42] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: evidence and implications," in *Proc. IEEE Inf. Commun. Conf. (INFOCOM)*, 1999, pp. 126–134.
- [43] E. Zeydan, E. Baştuğ, M. A. Kader, I. A. Karatepe, A. S. Er, and M. Debbah, "Big data caching for networking: moving from cloud to edge," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 36–42, Sep. 2016.
- [44] S. Gitzenis, G. S. Paschos, and L. Tassiulas, "Asymptotic laws for joint content replication and delivery in wireless networks," *IEEE Trans. Inf. Theory*, vol. 59, no. 5, pp. 2760–2776, May 2013.
- [45] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. MIT Press, 1998.
- [46] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3–4, pp. 279–292, May 1992.
- [47] S. Traverso et al., "Temporal locality in today's content caching: Why it matters and how to model it," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 5, pp. 5–12, Oct. 2013.