Deep Reinforcement Learning for Mobile Edge Caching: Review, New Features, and Open Issues

Hao Zhu, Yang Cao, Wei Wang, Tao Jiang, and Shi Jin

Abstract

Mobile edge caching is a promising technique to reduce network traffic and improve the quality of experience of mobile users. However, mobile edge caching is a challenging decision making problem with unknown future content popularity and complex network characteristics. In this article, we advocate the use of DRL to solve mobile edge caching problems by presenting an overview of recent works on mobile edge caching and DRL. We first examine the key issues in mobile edge caching and review the existing learning-based solutions proposed in the literature. We also discuss the unique features in the application of DRL in mobile edge caching, and illustrate an example of DRL-based mobile edge caching with trace-data-driven simulation results. This article concludes with a discussion of several open issues that call for substantial future research efforts.

Introduction

Cellular networks are undergoing substantial growth in data traffic as a consequence of the boom in versatile mobile devices and data-consuming application services like mobile video services [1]. According to recent estimates, the annual global mobile data traffic is growing at an exponential rate and will exceed half a zettabyte by 2021. It is challenging for cellular networks to guarantee the quality of experience (QoE) of users while keeping up with the significant growth in mobile data traffic. A promising approach to address this challenge is mobile edge caching, which is an efficient traffic localization technology. By caching popular contents at mobile network edges like macro base stations (MBSs), small base stations (SBSs), and even user equipments (UEs), user demands for the same content can be served locally. This effectively reduces the redundant data traffic and greatly improves the QoE of users [2].

The optimal cache content placement primarily depends on the content popularity distribution, which indicates what content will be requested in the future. Some popularity-based cache placement schemes assume that cache entities know the content popularity. However, as contents are favored by different users, local content popularity is influenced by the changing membership of the mobile users associated with the edge cache entity. Moreover, users' preferences in contents may vary in different contexts, including locations, network topologies, personal characteristics, and so on. Then future content popularity is actually unknown before making the caching decision.

Recently, the emergence of mobile edge computing (MEC) and fog computing empowers the edge network nodes with the capabilities of computation and storage, thus making it possible to achieve specific tasks like big data analytics and deep learning in edge networks. This allows the possibility of employing artificial intelligence (AI) techniques at mobile network edges for better understanding of mobile users' behaviors and network characteristics. By intelligently exploiting users' context information and statistical traffic patterns, a context-aware and intelligent mobile edge caching scheme can be designed. *Context awareness* enables each cache host to be aware of its operating environment, while *intelligence* enables each cache host to make the right decisions when selecting appropriate contents to be cached in the limited storage space at the right time for maximizing the caching performance.

As one of the AI techniques, reinforcement learning (RL) enables agents to deal with decision making problems by learning through interactions with the environment. Agents should derive proper representations of the environment and use these to generalize past experiences to new situations with sufficiently low complexity. Therefore, the applicability of traditional RL is usually inherently limited to domains in which useful features can be handcrafted, or to domains with fully observed, low-dimensional state spaces. The advent of deep neural networks has made it possible for artificial neural networks to automatically learn compact low-dimensional representations from raw, high-dimensional data. By combining RL and deep learning, deep reinforcement learning (DRL) enables agents to optimize their control in an environment by automatically learning knowledge directly from raw, high-dimensional observations [3]. Google DeepMind has successfully adopted DRL on some games with quite good results. One of the driving forces behind DRL is the vision of creating systems that are capable of learning how to act toward the real world without a certain model.

Digital Object Identifier:
10.1109/MNET.2018.1800109

Hao Zhu, Yang Cao, Wei Wang, and Tao Jiang (corresponding author) are with Huazhong University of Science and Technology; Shi Jin is with Southeast University.

0890-8044/18/$25.00 © 2018 IEEE

In this article, we advocate the use of DRL to achieve context awareness and intelligence in mobile edge caching. In order to better understand the motivation and potential of applying DRL in mobile edge caching, this article presents a comprehensive analysis of recent research related to DRL in mobile edge caching. First, we introduce the preliminaries of edge caching, and summarize the key issues faced in achieving context awareness and intelligence. Second, we review the state-of-the-art learning-based mobile edge caching schemes. These schemes are divided into two kinds of approaches: *popularity-prediction-based* approaches and *reinforcement-learning-based* approaches. We illustrate how these two kinds of schemes work differently, and discuss their advantages and disadvantages. Third, after an overview of DRL algorithms, we discuss the application of the DRL approach in mobile edge caching. Specifically, DRL has the ability to exploit new features that are not widely applied in existing mobile edge caching schemes but have great potential for performance enhancement. Moreover, we present some trace-data-driven simulation results to validate the effectiveness of DRL-based mobile edge caching. Finally, we discuss several open issues of DRL-based mobile edge caching. All discussions are presented in a tutorial manner in order to establish a foundation for further research in this field.

Preliminaries and Key Issues

In this section, we introduce some preliminaries on mobile edge caching and the key issues in efficient mobile edge caching.

Preliminaries on Mobile Edge Caching

To describe the sketch of mobile edge caching, we briefly introduce caching places, content popularity, and caching policies and algorithms as answers to questions of where to cache, what to cache, and how to cache, respectively.

Caching Places: Popular contents are reused, asynchronously, by many users. Thus, caching popular contents close to users is an efficient method to reduce redundant network traffic. In traditional cellular systems, the content requested by users has to be fetched from cache servers of the content delivery network, which can be far away from mobile networks. Then caching contents into the mobile core network is implemented. However, the limited-capacity backhaul links may still be congested due to the duplicated transmissions for popular contents, leading to poor QoE for users [4]. To offload the backhaul traffic, caching contents at BSs is an efficient solution with the evolution of BSs and low-cost storage units. In the future fifth generation (5G) networks, device-to-device (D2D) communication will enable the storage units at UEs to be exploited for content delivery. In general, contents can be cached in edge nodes such as MBSs, SBSs, and UEs [2], as shown in Fig. 1.

Content Popularity: There are a wide range of contents, including videos, audio files, Internet of Things data, and so on. The number of contents available over the Internet is extremely huge. Not all of them need to be cached in edge nodes, especially considering the fact that the storage space of edge nodes is limited. To decide what to cache in edge nodes, the popularity of content should be taken into consideration. Local content popularity is an index indicating the probability of a content being requested by users in the vicinity of an edge node. Most current works on mobile edge caching assume that content popularity follows a static Zipf distribution [5]. But since the user groups associated with individual edge nodes are different and user preferences may change with time, content popularity is unknown in advance and may change in space and time [6].

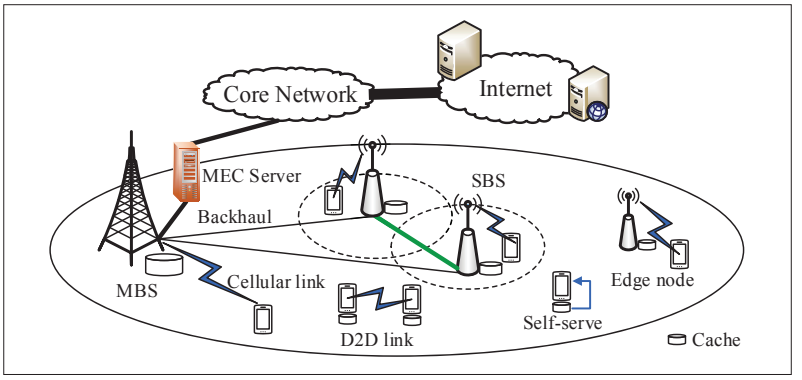

FIGURE 1. Architecture of mobile edge caching.

Caching Policies and Algorithms: Caching policies guide edge nodes to select cache contents with different aims such as traffic offloading, QoE improvement, and energy consumption reduction. These objectives are directly or indirectly related to the cache hit ratio, that is, the probability that the contents requested by users are cached in the edge networks. Conventional caching policies such as least frequently used (LFU) and least recently used (LRU) have been successfully adopted in wired caching, but could perform poorly in mobile edge caching due to being unable to carefully consider the specific characteristics of operating environments, for example, mobile network topology uncertainty, fading and interfered wireless channels, and user mobility. In the next subsection, we introduce the key issues faced in the design of efficient mobile edge caching policies and algorithms.

Key Issues: Context Awareness and Intelligence

An efficient mobile edge caching policy needs to be context-aware, and thus able to take environment factors that affect caching performance into consideration. Moreover, it needs to be intelligent to perceive the environment and then automatically take optimal or near-optimal caching actions as time goes by, without explicit programmed control rules. However, the operating environment of mobile edge caching is complex and dynamic. It is challenging to establish a context-aware and intelligent caching policy.

How to Establish a Context-Aware Caching Policy: For mobile edge caching, the operating environment is complex. For instance, local content popularity in the vicinity of an edge node is affected by a variety of context factors in a complex manner. Specifically, user preferences in terms of content are affected by user contexts, such as location, personal characteristics, and device diversity, in complicated patterns. Moreover, the edge node selected for serving a specific

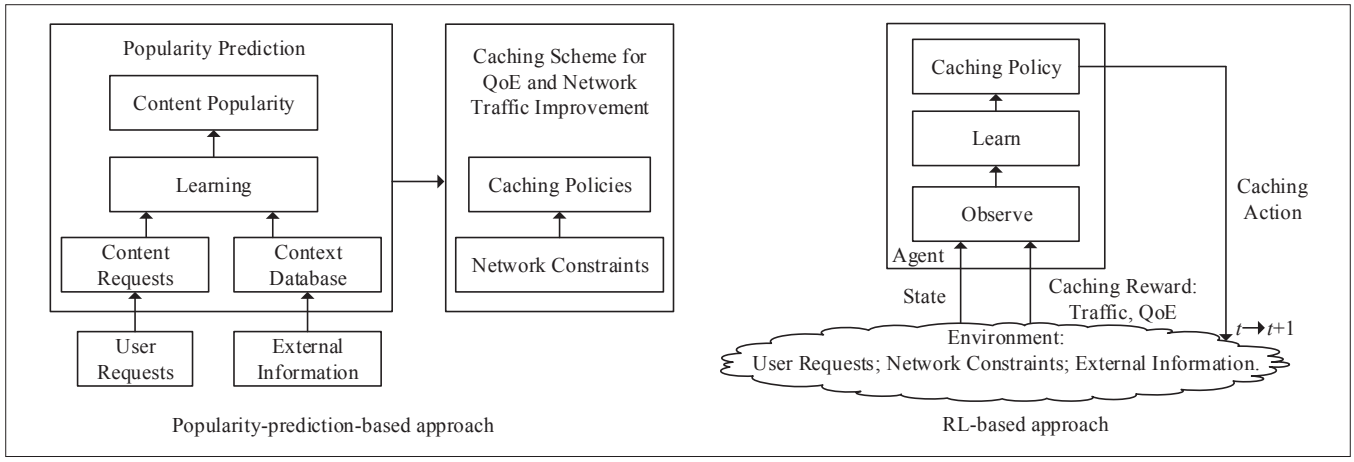

FIGURE 2. Schematic of learning-based mobile edge caching schemes.

user's request is impacted by network conditions, such as network topology, wireless channel, and cooperation among BSs, in sophisticated ways. The complex operating environment is difficult to model, thus making it difficult to tackle all factors that affect caching performance by rule and line. In other words, it is challenging to achieve context awareness in the design of mobile edge caching policies.

How to Establish an Intelligent Caching Policy: The operating environment of mobile edge caching evolves with time such that new states and actions may be encountered, due to the natural dynamics of wireless systems. Edge nodes should have the intelligence to learn new states and actions, as well as match them so that optimal or near-optimal actions can be taken. The adage "practice makes perfect" describes the concept of intelligence. Intelligence enables the edge node to learn about its actions by evaluating feedback, that is, the consequences of executing its actions. Then an intelligent caching policy should be receptive to the feedback, thus being capable of catering for the dynamics of the operating environment.

To achieve context awareness and intelligence in mobile edge caching, it is crucial to give edge nodes the ability to "learn" with data. Thus, edge nodes are able to make data-driven predictions on the operating environment, and make forward-looking caching decisions to maximize the long-term caching performance.

Classification of Learning-Based Mobile Edge Caching

In this section, we categorize the state of the art on learning-based mobile edge caching into two approaches, the popularity-prediction-based approach and the reinforcement-learning-based approach.

Popularity-Prediction-Based Approach

Many learning-based mobile edge caching schemes try to first predict content popularity, and then use the popularity estimations to devise caching polices, as shown on the left of Fig. 2. Diverse kinds of information such as statistical traffic patterns and context information are exploited to predict content popularity in wireless networks with or without assumptions of a priori knowledge of the popularity distribution. In [2], content popularity was predicted through collaborative filtering by exploiting users–file correlations and users' social relationships. In [5], time-varying popularity was estimated based on user requests and the content age by assuming an a priori model, called the Poisson shot noise model, for the popularity distribution. Online learning was employed in [6] to estimate content popularity by learning the access patterns of different contents. In [7], content popularity was predicted by taking the interest and behavior of users into consideration, based on the Indian Buffet model. An extreme learning machine was used in [8] to estimate the content popularity based on users' behavior, features of content, and request statistics. After the procedure of popularity prediction, the caching policies and algorithms can be obtained by solving optimization problems with network constraints or revising traditional content caching algorithms with estimated popularity. For example, in [8], a mixed integer linear programming problem was constructed to perform cache initialization, and LRU was revised as a segmented LRU with three segments (S3LRU) cache replacement policy for dynamically adjusting the cache based on the predicted popularity. In this kind of approach, even though an accurate prediction on the content popularity is obtained, it is still difficult to model realistic edge networks, which are complex and dynamic, and it is also challenging to solve the caching placement problems, which are usually NP-hard.

Reinforcement-Learning-Based Approach

Another category for learning-based mobile edge caching schemes is the RL-based approach. Instead of separating learning and content placement, RL treats them as a whole, as shown on the right of Fig. 2. The caching policy adopted by the RL agent could be trained with observations, solely based on a reward signal resulting from its actions, rather than tackling every single factor that affects caching performance such as network topology and nodal mobility. This reward may be offloaded traffic or QoE, which covers a wide range of factors that can affect the performance. Different RL algorithms have been applied in the design of mobile edge caching schemes.

The problem of content placement in an SBS was solved based on the multi-armed bandit (MAB) theory in [9]. The problem of content caching and sharing among multiple cooperative SBSs was solved through the MAB approach in [10]. In [11], a distributed cache replacement policy for mobile edge caching at BSs with D2D offloading was proposed based on Q-learning. Note that traditional RL-based mobile edge caching schemes are limited to domains with fully observed, low-dimensional state spaces. However, the operating environment of edge caching is complex, and it is difficult to manually extract all useful features of the environment as low-dimensional state spaces. With the advent of DRL, agents could be directly trained on raw, high-dimensional observations, rather than handcrafting useful features or low-dimensional state spaces. In the next section, we introduce the application of DRL in mobile edge caching.

Application of DRL in Mobile Edge Caching

Before the introduction of the application of DRL in mobile edge caching, a brief review of RL and DRL is first described. For a more comprehensive survey of RL and DRL, we refer readers to the overview in [3].

Reinforcement Learning

An RL agent learns through interacting with its environment over time. At each time step t, the agent observes a state s_t in a state space $\mathcal{S}$ about its environment, and chooses an action a_t from an action space $\mathcal{A}$ following a behavior policy $\pi = P(a_t | s_t)$, which is a mapping from state s_t to a probability of choosing action a_t. Then the agent obtains a reward r_t and transitions to a new state s_{t+1}, according to the environment dynamics, or model, for reward function $\mathcal{R}(s, a)$ and state transition probability $P(s_{t+1} | s_t, a_t)$ respectively. The accumulated reward is defined as return $R_t = \Sigma_{k=0}^{\infty} \gamma^k r_{t+k}$ with a discount factor $\gamma \in (0,1]$.

The goal of the agent is to find an optimal policy, π^*, which achieves the maximum expected return from all states. The state-value function $V^{\pi}(s) = E[R_t | s_t = s]$ and the action-value function $Q^{\pi}(s, a) = E[R_t | s_t = s, a_t = a]$ can measure how good π is. $V^{\pi}(s)$ represents the expected return for following policy π from state s, and $Q^{\pi}(s, a)$ represents the expected return for selecting initial action a in state s and then following π. The advantage function $A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$ represents the relative advantage of actions.

There are two kinds of methods to solve RL problems: methods based on value functions and methods based on policy search. In value-function-based methods, the most important task is to properly and efficiently estimate the value function. With the improvement of the estimate, the policy can naturally be improved by selecting actions greedily according to the updated value function. Rather than maintain a value function model, policy-search-based methods search directly for an optimal policy by parameterizing the policy. The parameterized policy is denoted by π_θ, whose parameters θ are updated to maximize the expected return $E[R | \theta]$ using either gradient-based or gradient-free optimization. There is also a hybrid, actor-critic approach, where both value functions and policy search are adopted.

Exploration means taking perceived non-optimal actions with the aim of finding better actions, when the policy is not optimal yet or the environment is not entirely explored. An example of this is the ε-greedy exploration policy.

By combining value functions with an explicit representation of the policy, the "actor" (policy) can learn based on the feedback from the "critic" (value function).

In all RL methods, it is critical for RL agents to trade off exploitation and exploration. Exploitation means taking the perceived best action to maximize rewards greedily. Exploration means taking perceived non-optimal actions with the aim of finding better actions when the policy is not optimal yet or the environment is not entirely explored. An example of this is the ε-greedy exploration policy, which selects a random action with probability $\varepsilon \in [0,1]$, and the optimal action otherwise.

Deep Reinforcement Learning

Unlike traditional RL, DRL can efficiently deal with the curse of dimensionality by utilizing the advantages of deep neural networks on automatically learning low-dimensional feature representations. In general, DRL is based on training a deep neural network to approximate the optimal policy π^*, and/or the optimal value function V^* or Q^*.

In [12], the authors proposed Deep Q-Network (DQN) and ignited the field of DRL. The deep convolutional neural network is used to represent the action-value function, denoted as $Q(s, a; \theta)$. The parameters θ are the weights of the neural network, and the Q-network is trained by updating θ at each iteration with the loss function being set as the mean square error of difference between the action-values $Q(s, a; \theta)$ and the target values:

$$r + \gamma \max_{a'} Q(s', a'; \theta).$$

To address the instabilities caused by approximating the action-value function with neural networks, experience replay and target networks are employed in DQN. Experience replay, which randomizes over the data, is able to remove correlations in the observation sequence and smooth over changes in the data distribution. Target networks, which represent the target Q values and are only periodically updated, are able to reduce the correlations between the action-values and the target values.

There are many extensions of DQN that try to improve the function approximator for the Q-function. Double-Q learning achieves a better estimate by employing a double estimator for tackling the over-estimate problem [13]. Rather than having to estimate the Q-value for each available action, the duelling DQN benefits from a single baseline for the state in the form of V^{π}, and easier-to-learn relative values in the form of A^{π} [14].

Deep neural networks can also be employed to represent the policy directly. In the following, we focus on asynchronous advantage actor-critic (A3C) [15], which maintains a policy $\pi(a | s; \theta)$ (actor network) and an estimate of the value function $V(s; \theta_v)$ (critic network). A3C combines

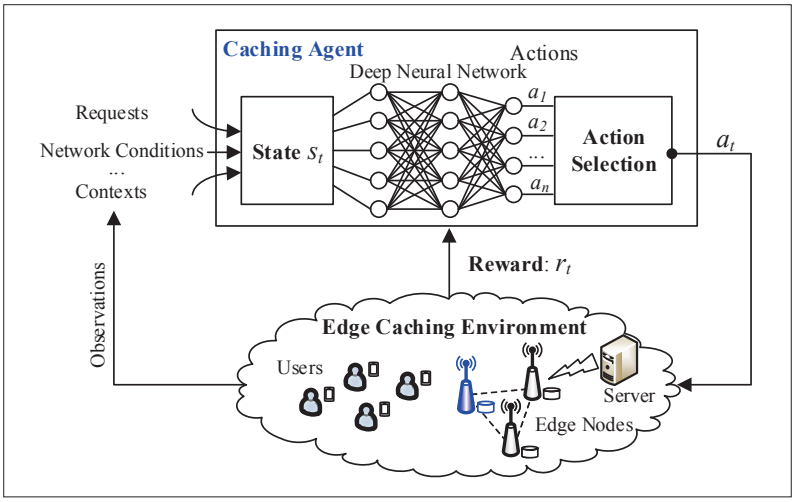

FIGURE 3. Applying DRL to mobile edge caching.

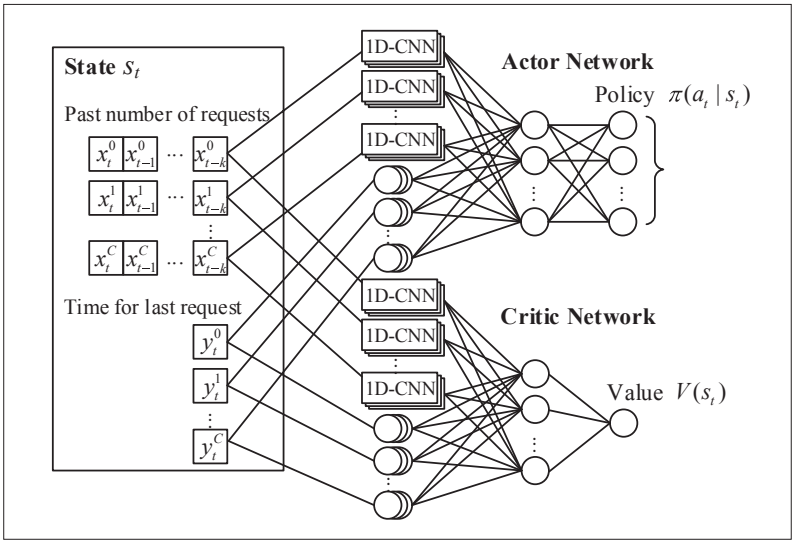

FIGURE 4. An example of DRL-based mobile edge caching.

advantage updates with the actor-critic formulation, and relies on asynchronously updated policy and value function networks trained in parallel over several processing threads. The gradient of the expected return with respect to actor network parameters, θ, is computed as $\nabla_\theta E[R_t \mid \theta] = E[\nabla_\theta \log\pi(a_t \mid s_t; \theta) A(s_t, a_t; \theta, \theta_v)]$, where $A(s_t, a_t; \theta, \theta_v)$ can be estimated as $r_t + \gamma V(s_{t+1}; \theta_v) - V(s_t; \theta_v)$. As for critic network parameters, they can be trained according to the temporal difference method, where the loss function is set as the mean square error of the difference between the state-values $V(s_t; \theta_v)$ and the target values $r_t + \gamma V(s_{t+1}; \theta_v)$.

DRL-Based Mobile Edge Caching

DRL can be applied in mobile edge caching, and is able to learn caching policy automatically, without using any pre-programmed control rules or explicit assumptions about the operating environment. Figure 3 summarizes the components and features of DRL-based mobile edge caching, namely high-dimensional state space representation, neural network structure design, and long-term reward maximization. As shown, the caching agent observes the environment and obtains serval raw signals, such as user requests, context information, and network conditions. These signals can be assembled into a high-dimensional state input and then fed to the deep neural network. The deep neural network needs to be designed into a specific structure like a convolutional neural network or a recurrent neural network, which is able to mine useful information and output the value function or the policy. According to the output, an action, which represents the contents cached at the next slot, can be selected. The resulting caching performance is then observed and passed back to the caching agent as a reward. The caching agent uses the reward to train and improve its deep neural network model with the aim of maximizing the expected accumulated discount reward.

We illustrate a simple example of applying DRL to the mobile edge caching, where one edge node is considered. All contents are assumed to have the same size, and the storage space of the edge node is assumed to be enough for caching C contents. The edge node can serve each request t directly if the requested content has been cached locally. Otherwise, the edge node requests this content from the original server and updates the local cache according to the caching policy. Our aim is to find the optimal caching policy maximizing the offloaded traffic, that is, the number of contents answered by the edge node. This problem can be solved based on A3C, which involves training an actor network for representing the policy and a critic network for estimating the state-value function. The detailed functionalities of these networks are shown in Fig. 4 and explained below.

Input: After each request t, the caching agent takes state inputs

$$s_t = \left(\vec{x}_t^0, \vec{x}_t^1, \ldots, \vec{x}_t^C, \vec{y}_t^0, \vec{y}_t^1, \ldots, \vec{y}_t^C\right)$$

to its neural networks. We only extract features from the currently requested content, whose index is 0, and the cached contents, whose indexes range from 1 to the cache size C. $\vec{x}_t^i$ is a vector that represents the number of requests for content i within the the past k groups of requests, where each group consists of T requests. y_t^i denotes the time when content i was most recently requested. Actually, besides the request information stated here, more raw observations on contexts and edge networks can also be included in the state inputs.

Policy: Upon receiving s_t, the caching agent needs to take an action a_t that corresponds to whether or not to cache the currently requested content in the cache, and if yes, the agent determines which local content will be replaced. The agent selects actions based on the policy $\pi(a \mid s; \theta)$, which is represented by the actor network.

Policy Gradient Training: After applying each action, the mobile edge caching environment provides the caching agent with a reward r_t, which is defined as the offloaded traffic in each request. Each update of the actor network parameters θ follows the policy gradient

$$\theta \leftarrow \theta + \alpha \sum_t \nabla_\theta \log \pi(a_t \mid s_t; \theta) A(s_t, a_t; \theta, \theta_v), \quad (1)$$

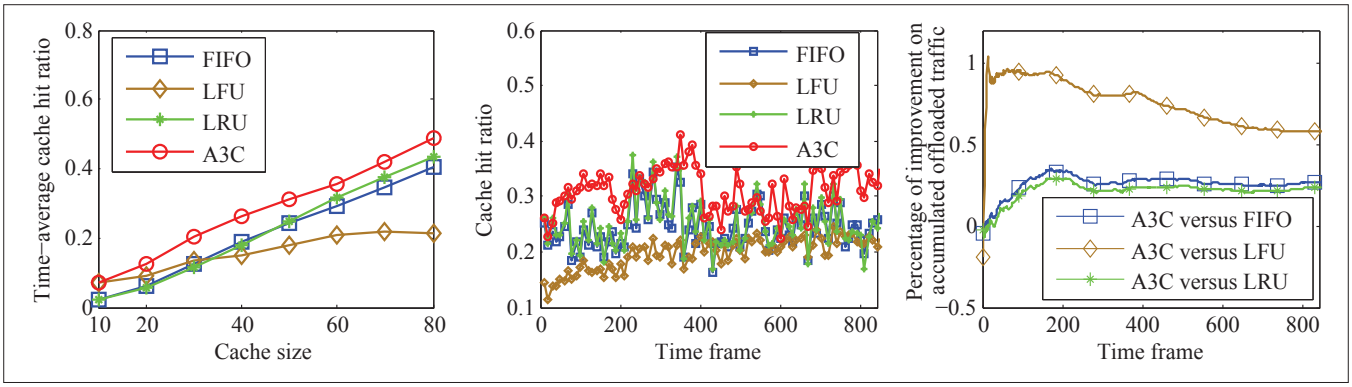

FIGURE 5. Performance comparison between different caching policies.

and the update of the critic network parameters θ_v follows the temporal difference method

$$\theta_v \leftarrow \theta_v - \alpha' \sum_t \nabla_{\theta_v} (r_t + \gamma V(s_{t+1}; \theta_v) - V(s_t; \theta_v))^2, \quad (2)$$

where α and α' are the learning rates. Equation 1 takes an update step in direction $\nabla_\theta \log \pi(a_t | s_t; \theta)$, which aims at increasing $\pi(a_t | s_t; \theta)$ (the probability of selecting action a_t in state s_t). The size of the step is determined by the value of the advantage for action a_t in state s_t. To trade off exploration and exploitation, an entropy regularization term, which is defined as a weighted gradient of the entropy of the policy (the probability distribution over actions) at each time step, can be added into Eq. 1. The training process can be offline or online. Through the offline way, the caching policy is generated a priori (during a training phase) and then stays unchanged after deployment. Through the online way, the caching policy is directly trained on edge nodes and periodically updated as new data arrives.

Data-Driven Simulation Results

We use the real-world MovieLens data (https://grouplens.org/datasets/movielens), which includes 20 million ratings applied to 27,000 movies by 138,000 users, to validate the efficiency of the A3C-based mobile edge caching policy. Similar to [6], we assume that every movie rating is a video request since the rating data exhibits similar access patterns to those of content request data. More specifically, we assume that every movie comment in our dataset is a downloading/streaming request, and the time when the comment is posted is considered as the time when the request is initiated. In the simulation, the number of past groups k is set as 6, each of which consists of T = 100 requests. The past number of requests for each considered content is passed to a one-dimensional convolution layer (1D-CNN) with 64 filters, each of size 2 with stride 1. Results from these layers and other inputs are then aggregated together in a hidden layer with 64 neurons. The last layer of the actor network implements the softmax function to output the policy. The critic network applies the same neural network structure, but its final output is a linear neuron. Online training is adopted to update these networks.

We evaluate the performance of the proposed policy on cache hit ratio and offloaded traffic. Three common baseline caching policies are considered: LFU, LRU, and first in first out (FIFO). The results are shown in Fig. 5. On the left of Fig. 5, we can see that our proposed policy achieves the highest average cache hit ratio for all cache capacity values. The middle of Fig. 5 shows the fluctuation of cache hit ratio as time goes by, with a fixed cache size C = 50. Since new movies are generated as time goes by and the popularity of each movie varies with time, the distribution of content requests is unstable. However, the proposed policy is able to maintain its advantage over other baselines in terms of cache hit ratio among almost all timeframes. The right side of Fig. 5 shows the percentage of improvement on accumulated offloaded traffic when the proposed policy is compared to three other baselines. We observe that the percentage is negative at first, and then evolves and converges to a positive value. This means that the proposed policy outperforms other baselines if efficient training of the neural network is guaranteed.

Open Issues

This section discusses several open issues that can be pursued so that the application of DRL in mobile edge caching may achieve better performance.

Incorporation of Deep Models into DRL-Based Mobile Edge Caching

By incorporating with a transition model that represents the state transition probability distribution, the DRL agent is able to reduce the amount of required interactions with the real environment and speed up its learning. The reason is that the transition model allows for simulation of the environment without direct interactions with the environment. Predictive models of the dynamic mobile edge caching environment can be obtained through deep learning. For example, a deep dynamic model can be established by using autoencoders to embed high-dimensional observations into a lower-dimensional space. Deep-neural-network-based transition models are able to deal with some of the problems caused by using imperfect models. Specifically, the activations and outputs of these models are embedded into a vector; thus, DRL agents are able to obtain more information than just the final result of any model rollouts. Besides, DRL agents can also learn to downplay the information when they think the model results are incorrect.

It is possible to exploit previously acquired knowledge from other domains to speed up learning in DRL. For example, knowledge about the content distribution pattern in social networks can be utilized by the caching agent in D2D-enabled edge networks.

Cooperation between Multiple Caching Agents

Multiple edge nodes may have overlapping coverage, where each request only needs to be served by only one edge node. Moreover, cached contents can be shared between neighboring edge nodes. Cooperation is critical for multiple edge nodes to make optimal caching decisions and then avoid reduplicative caching. In DRL-based mobile edge caching, it is important to design efficient methods for passing messages between agents, since the information exchanged may become stale as time goes by. According to where the caching actions of multiple agents are decided, cooperative edge caching schemes can be divided into centralized approaches and distributed approaches. In centralized approaches, a global reward of the edge network can be taken into consideration. In distributed approaches, each agent has a local reward, and the action of a caching agent may change the environment experienced by another agent. Then multiple equilibria will be a challenging issue.

Utilization of Knowledge from Other Domains

It is possible to exploit previously acquired knowledge from other domains to speed up learning in DRL. For example, knowledge about the content distribution pattern in social networks can be utilized by the caching agent in D2D-enabled edge networks. The combination of transfer learning and DRL has advantages in constructing more efficient caching policies. On one hand, transfer learning can be used to help train features that can be used by DRL agents, thus making them easier to converge to the optimal caching policy. On the other hand, transfer learning can be applied to transfer policies learned by other relative networks to the DRL-based caching agents, thus improving the sample efficiency and robustness of current DRL algorithms.

Adaptation to Dynamics of the Environment

In mobile edge networks, the environment may change abruptly. For instance, edge nodes or users may move in random directions at different speeds; thus, the network topology may change slowly or quickly. To cater for the dynamics of the environment, the agent should be able to perceive the underlying patterns of environments' dynamics through the neural network from observations. Besides, the trade-off between exploration and exploitation may also need to be adjusted. Taking perceived non-optimal actions (exploration) may be encouraged when a new situation is encountered, and exploiting the perceived optimal action may be encouraged if the environment has been well explored.

Conclusions

In this article, we advocate the use of DRL to achieve context awareness and intelligence in mobile edge caching. We first introduce the background of mobile edge caching and the key issues. Existing learning-based solutions have also been reviewed and categorized. We apply DRL into mobile edge caching, thus making caching agents able to cater for complex and dynamic environments by learning through raw observation signals. An example has been illustrated with trace-data driven simulation results. Finally, we discuss several open issues.

Acknowledgment

This work was supported in part by the National Science Foundation of China with Grant numbers 61729101, 61601193, 61720106001, 61871441, 61502114, 91738202, and KF20181911, the Major Program of the National Natural Science Foundation of Hubei in China with Grant 2016CFA009, and the Fundamental Research Funds for the Central Universities with Grant number 2015ZDTD012.

References

[1] J. Wu *et al.*, "Content-Aware Concurrent Multipath Transfer for High-Definition Video Streaming over Heterogeneous Wireless Networks," *IEEE Trans. Parallel Distrib. Sys.*, vol. 27, no. 3, Mar. 2016, pp. 710–23.
[2] E. Bastug, M. Bennis, and M. Debbah, "Living on the Edge: The Role of Proactive Caching in 5G Wireless Networks," *IEEE Commun. Mag.*, vol. 52, no. 8, Aug. 2014, pp. 82–89.
[3] K. Arulkumaran *et al.*, "Deep Reinforcement Learning: A Brief Survey," *IEEE Signal Processing Mag.*, vol. 34, no. 6, Nov. 2017, pp. 26–38.
[4] K. Zhang *et al.*, "Cooperative Content Caching in 5G Networks with Mobile Edge Computing," *IEEE Wireless Commun.*, vol. 25, no. 3, June 2018, pp. 80–87.
[5] M. Leconte *et al.*, "Placing Dynamic Content in Caches with Small Population," *Proc. IEEE INFOCOM*, San Francisco, CA, July 2016.
[6] S. Li *et al.*, "Trend-Aware Video Caching through Online Learning," *IEEE Trans. Multimedia*, vol. 18, no. 12, Dec. 2016, pp. 2503–16.
[7] X. Zhang *et al.*, "Information Caching Strategy for Cyber Social Computing Based Wireless Networks," *IEEE Trans. Emerging Topics Comp.*, vol. 5, no.3, July–Sept. 2017, pp. 391–402.
[8] S. Tanzil, W. Hoiles, and V. Krishnamurthy, "Adaptive Scheme for Caching YouTube Content in a Cellular Network: Machine Learning Approach," *IEEE Access*, vol. 5, 2017, pp. 5870–81.
[9] P. Blasco and D. Gunduz, "Learning-Based Optimization of Cache Content in a Small Cell Base Station," *Proc. IEEE ICC*, Sydney, Australia, June 2014.
[10] J. Song et al., "Learning-Based Content Caching and Sharing for Wireless Networks," *IEEE Trans. Commun.*, vol. 65, no. 10, Oct. 2017, pp. 4309–24.
[11] W. Wang, *et al.*, "Edge Caching at Base Stations with Device-to-Device Offloading," *IEEE Access*, vol. 5, 2017, pp. 6399–6410.
[12] V. Mnih *et al.*, "Human-Level Control through Deep Reinforcement Learning," *Nature*, vol. 518, no. 7540, Feb. 2015, pp. 529–33.
[13] H. Van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," *Proc. AAAI*, Phoenix, AZ, Feb. 2016.
[14] Z. Wang *et al.*, "Dueling Network Architectures for Deep Reinforcement Learning," *Proc. ICML*, NY, June 2016.
[15] V. Mnih *et al.*, "Asynchronous Methods for Deep Reinforcement Learning," *Proc. ICML*, NY, June 2016.

Biographies

Hao Zhu has been a Ph.D. candidate at the School of Electronics Information and Communications, Huazhong University of Science and Technology, Wuhan, P.R. China, since 2014. He received his B.S degree in information and communication engineering from Huazhong University of Science and Technology in 2014. His research interests lie in device-to-device communications, mobile edge networks, and multimedia communications.

Yang Cao [S'09, M'14] is currently an associate professor with the Wuhan National Laboratory for Optoelectronics, School of Electronics Information and Communications, Huazhong University of Science and Technology, where he received his B.S.

and Ph.D. degrees. From 2011 to 2013, he was a visiting scholar with the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe. His research interests include 5G cellular networks, the Internet of Things, and future networks. He received the CHINACOM Best Paper Award in 2010 and a Microsoft Research Fellowship in 2011.

Wei Wang [S'10, M'16] is currently a professor with the Wuhan National Laboratory for Optoelectronics, School of Electronics Information and Communications, Huazhong University of Science and Technology. He received his Ph.D. degree from the Department of Computer Science and Engineering at Hong Kong University of Science and Technology. He has served as a Guest Editor of *Wireless Communications and Mobile Computing*, on the IEEE ComSoc MMTC Communications, and as TPC of INFOCOM, GLOBECOM, and so on. His research interests include PHY/MAC designs, security, and mobile computing in cyber-physical systems.

Tao Jiang [M'06, SM'10] is currently a Chair Professor with the Wuhan National Laboratory for Optoelectronics, School of Electronics Information and Communications, Huazhong University of Science and Technology. He has authored or co-authored more than 300 technical papers in major journals and conferences and five books in the areas of wireless communications and networks. He was invited to serve as TPC Symposium Chair for IEEE GLOBECOM 2013, IEEEE WCNC 2013, and ICCC 2013. He has served or is serving as Associate Editor of some technical journals in communications, including *IEEE Transactions on Signal Processing*, *IEEE Communications Surveys & Tutorials*, *IEEE Transactions on Vehicular Technology*, and the *IEEE Internet of Things Journal*, and is the Associate Editor-in-Chief of *China Communications*.

Shi Jin [M'07, SM'17] is a professor in the faculty of the National Mobile Communications Research Laboratory, Southeast University. His research interests include 5G and beyond, random matrix theory, and information theory. He serves as an Associate Editor for *IEEE Transactions on Wireless Communications*. He was awarded the 2011 IEEE Communications Society Stephen O. Rice Prize Paper Award in the field of communication theory and the 2016 IEEE GLOBECOM Best Paper Award.