Hierarchical Reinforcement Learning-based Mobility-aware Content Caching and Delivery Policy for Vehicle Networks [⋆]

Le Zhang[1], Yongxuan Lai[1,3](✉), and Fan Yang[2]

[1] School of Informatics / Shenzhen Research Institute, Xiamen University, Xiamen 361005, China
zhangle@stu.xmu.edu.cn, laiyx@xmu.edu.cn
[2] School of Aerospace Engineering, Xiamen University, Xiamen 361005, China
yang@xmu.edu.cn
[3] School of Mathematics and Information Engineering, Longyan University, Longyan 364012, China

Abstract. Mobile Edge Computing (MEC) has been regarded as a promising technology to satisfy the growing demand for resource-intensive applications in vehicle networks. Content caching and delivery, a critical problem in MEC, has attracted much research attention in the past decade. However, most existing caching schemes in the vehicle network scenario still confront two challenges: 1) High mobility of vehicles results in unstable connectivity; 2) Fairly massive state spaces of existing schemes have become their obstacles to good scalability. To address these challenges, we propose a hierarchical reinforcement learning (HRL)-based mobility-aware content caching and delivery policy for vehicle networks. First of all, we formulate the caching and delivery problem as a Markov decision process (MDP) problem. Our aim is to minimize the time-averaged transmission cost in the vehicle network scenario. To address the curse of dimensionality, we decompose the joint optimization of content caching and delivery into the vehicle side and RSU side subproblems. DDPG and Double-DQN are applied to address these two subtasks. Furthermore, an LSTM-based location prediction module is built to mine the mobility patterns of vehicles. Experimental studies and analysis, which are conducted on a real-world dataset, demonstrate that our approach outperforms other baseline schemes in terms of transmission cost and convergence speed.

Keywords: mobile edge computing · vehicle network · content caching and delivery · hierarchical reinforcement learning · deep deterministic policy gradient.

[⋆] This work is partially supported by Longyan Industry-Education Integration Project of Xiamen University (20210302) and the Natural Science Foundation of Guangdong (2021A1515011578).

1 Introduction

The past decade has witnessed the widespread adoption of smart vehicles across Intelligent Transportation Systems (ITS). Meanwhile, the wide commercial rollout of fifth-generation (5G) networks has facilitated a wide range of innovative vehicle applications, such as in-car entertainment, autonomous driving and live traffic monitoring [9]. However, there are some challenges in providing these applications with high quality of experience (QoE) in Vehicle Networks (VN). First of all, the emerging content will consume an extremely large data volume, such as video streaming and Virtual Reality/Augmented Reality (VR/AR). Secondly, many multimedia applications also involve live interaction between users, which requires low-latency content delivery [23]. These challenges are difficult to solve in the traditional centralized cloud-based computation. Fortunately, Mobile Edge Computing (MEC) has emerged as a promising paradigm to support these resource-demanding applications with stringent latency and reliability requirements. MEC moves computing and storage resources to close proximity to mobile users, such as Base Stations (BSs) and Road Side Units (RSUs), so that the latency to users will be remarkably reduced [1, 4]. At the same time, cache nodes at the edge of networks can cache popular contents in advance, which alleviates strain on the backhaul links.

Content caching and delivery is a critical problem in mobile edge computing, which has attracted much research attention in recent years. Generally speaking, the storage space of edge nodes is often limited, which necessitates the careful design of an efficient content caching and delivery policy. Most existing caching schemes can be classified into two categories: reactive caching and proactive caching [8]. Firstly, Least Recently Used (LRU), Least Frequently Used (LFU), and their variants are examples of reactive caching. These caching schemes are driven by simple statistics of request history and mine user request patterns to guide the cache decision. They are easy to implement and widely used in Content Delivery Networks (CDNs). However, they are not suitable for the features of vehicle network scenarios, such as high mobility and dynamic content popularity. In contrast, proactive caching schemes predict content popularity in advance and cache the most popular contents which are possible to be accessed in the recent future. In proactive caching schemes, it is of vital importance to accurately predict content popularity. A large number of existing studies in the broader literature have investigated machine learning-based proactive caching schemes by utilizing deep learning [13,19], reinforcement learning [12,15,23] and federated learning [22], etc.

The advent of autonomous driving and demand for improved road safety and in-car entertainment have led to the development of vehicle-to-everything (V2X) technology and vehicle networks (VN). The integration of vehicle network and MEC will greatly promote the sensing and computing capability of vehicle network at the network edge. Despite recent advancements in Machine Learning(ML)-based proactive caching, utilizing ML approaches for edge caching in vehicle networks still confronts the following two challenges: 1) High mobility: High movement of vehicles results in unstable connectivity, and vehicles may not have enough time to download the entire requested content during the time staying in the area of one edge node. In order to improve the quality of experience, cache schemes must be mobility aware and enable multiple cache nodes to collaborate; 2) Scalability: Most of existing studies use end-to-end reinforcement learning techniques, which makes their state spaces very large and impairs their potential to scale. Specifically, the state space expands exponentially as the number of linked vehicles rises, delaying the model's convergence. This is so-called *the curse of dimensionality* [2].

In past decades, Hierarchical Reinforcement Learning (HRL) has been regarded as one of the promising technologies to alleviate the curse of dimensionality and scale reinforcement learning to the long-horizon tasks [10]. HRL decomposes a long-horizon(i.e., large state and action space) reinforcement learning task into a hierarchy of subproblems or subtasks. This inspired us to come up with our own solution for content caching and delivery for vehicle networks. Different from the existing schemes of HRL-based caching [7, 11], our scheme focuses on the edge caching in vehicle network scenarios and applies a divide-and-conquer strategy to reduce the dimensions. In this paper, we propose a hierarchical reinforcement learning(HRL)-based mobility-aware content caching and delivery policy for vehicle networks, which has better scalability and universality. This caching scheme aims to minimize the time-averaged transmission cost in the scenario of vehicle networks with high mobility and dynamic content popularity. The content caching and delivery problem is formulated as an MDP problem and decomposed into vehicle side and RSU side subproblems. Deep deterministic policy gradient (DDPG) and double deep Q-network (Double-DQN) are applied to solve these two subproblems. The major contributions of this paper are as follows:

- We propose an HRL-based mobility-aware content caching and delivery policy for vehicle networks with high mobility and dynamic content popularity. The joint optimization of content caching and delivery is modeled as a Markov decision process (MDP) problem. This framework aims to minimize the time-averaged transmission cost in the scenario of vehicle networks.
- Borrowing ideas from HRL, we decompose the joint optimization of content caching and delivery into vehicle side and RSU side subproblems. DDPG and Double-DQN are used to address these two subtasks. Furthermore, a Long Short Term Memory(LSTM)-based location prediction module is built to mine the mobility patterns of vehicles. Through these efforts, we hope to achieve better scalability and generality.
- Experimental studies and analysis are conducted on a real-world taxi trajectory dataset, which demonstrates that our approach outperforms other baseline schemes in terms of transmission cost and convergence speed.

The rest of this paper is organized as follows. Section 2 summarizes the related work of this paper. Section 3 presents the system model. Section 4 formulates the content caching and delivery problem and reformulates it as an MDP problem. In Section 5, we introduce the detailed hierarchical reinforcement learning-based mobility-aware content caching and delivery policy for vehicle networks. Experimental studies and analysis are provided in Section 6. Finally, Section 7 concludes the paper and presents some future directions.

2 Related Work

Content Caching and Delivery in MEC. Mobile edge computing provides cloud computing and caching capabilities at the edge of cellular networks. A large number of recent works have been reported to investigate content caching and delivery in mobile edge networks. Jiang et al. [5] proposed a cooperative content caching and delivery framework to minimize average content downloading latency. The content caching problem is formulated as an integer-linear programming problem and solved by using the subgradient method. The content delivery policy is formulated as an unbalanced assignment problem and solved by using Hungarian algorithm. Yao and Ansari [21] jointly optimized the content placement and storage allocation for Internet of Things (IoT) to minimize the total network traffic cost. Two heuristic algorithms were proposed in order to reduce the computational complexity of the problem. Sun et al. [16] designed a cooperative content caching approach among small cells, for which the tradeoff between content delivery latency and storage cost was investigated.

Content Caching and Delivery in Vehicle Networks. The vehicle network, which is a promising paradigm to support diverse vehicular applications, has drawn much research attention with a wide range of works on content caching and delivery. In [3], a vehicle-based distributed storage scheme via local vehicle-to-vehicle (V2V) communications was proposed to cope with the vehicle mobility issue. Structured redundancy via erasure coding was also introduced in order to combat the volatile V2V links. In order to deal with the challenges of high mobility and privacy, Yu et al. [22] proposed a mobility-aware proactive edge caching scheme based on federated learning to leverage the private training data distributed on local vehicles for predicting content popularity. Context-aware adversarial autoencoder (C-AAE) was introduced to predict the highly dynamic content popularity.

Content Caching and Delivery with DRL. In the past decade, with the continuous enhancement of computing power brought by graphics processing units (GPUs), deep learning (DL) has aroused great interest and extensive research with fruitful outcomes in academia. Deep reinforcement learning (DRL), which combines the advantages of deep learning and reinforcement learning, is regarded as a powerful tool to solve sequential decision making problems. As a result, much effort has been made to utilize DRL to deal with content caching and delivery in vehicle networks. Qiao et al. [12] formulated the joint content caching and delivery optimization problem as a double timescale Markov decision process (DTS-MDP), and deep deterministic policy gradient (DDPG) was leveraged to obtain a suboptimal solution. However, its state space was fairly large and the method did not have good scalability. In [23], Zong et al. employed an ensemble of constituent caching policies and a DRL agent was trained to adaptively select the best-performing policy to control the cache. But the single cache node was its downside and the migration to the vehicle network scenario was difficult as a result. In [15], a QoE-driven edge caching method for the IoV was proposed to solve the RSU caching optimization problem. A class-based user interest model was established, which was more suitable for systems with a large number of small files. A deep reinforcement learning method was designed to address the QoE-driven RSU cache update issue effectively.

Most of the above-mentioned studies are end-to-end reinforcement learning approaches, which leads to their state spaces being fairly massive and having poor scalability. As the number of connected vehicles grows, the state space increases exponentially, which results in slow convergence of the model. A more systematic and theoretical analysis is required for the curse of dimensionality in content caching and delivery problem. This motivates us to propose a hierarchical reinforcement learning(HRL)-based mobility-aware content caching and delivery policy for vehicle networks, which has better scalability and universality.

3 System Model

In this section, we propose the content caching and delivery framework in vehicle networks including network model, communication model, mobility model and request model.

3.1 Network Model

We consider a general model of mobile edge computing in vehicle networks with different types of edge caching nodes, including a macro base station (MBS), several road side units (RSU) and content requesting vehicles (CRV), as shown in Figure 1. Let $\mathbb{N} = \{0, 1, ..., N\}$ represent the index set of edge caching nodes, in which 0 is the index of MBS and $\{1, 2, ..., N\}$ is the index set of RSUs. Let $\mathbb{K} = \{1, 2, ..., K\}$ denote the index set of CRVs to make requests to access contents. The index set of all available contents is denoted by $\mathbb{F} = \{1, 2, ..., F\}$. We assume that the MBS is the centralized content provider and has the abundant storage capacity to cache all available contents. Furthermore, each RSU $n \in \mathbb{N}$ and each CRV $k \in \mathbb{K}$ are equipped with a limited caching storage capacity, which is represented by M_n and L_k respectively.

The MBS serves all the RSUs with all the contents and the connections between the MBS and RSUs use optical fibers. Vehicles traverse the coverage areas of several RSUs and each CRV communicates with only one RSU through wireless links at the same time. The system runs over an infinite period of time, which is divided into slots, denoted as $t = 0, 1, 2, ...$. During the content caching and delivery process, a CRV will require a desired content (e.g., navigation map update, video streaming, etc.). Then, the local cache of CRV will be checked first

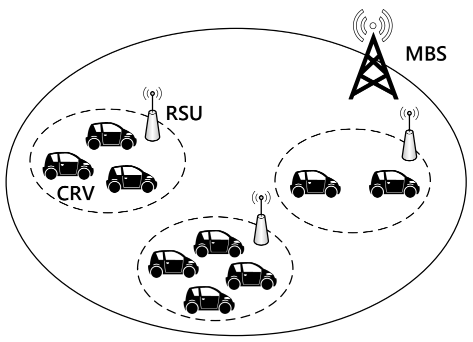


Fig. 1. The network model consists of a macro base station (MBS), several road side units (RSU) and content requesting vehicles (CRV).

to see whether the requested content is cached. The cache state of each content for CRV k is denoted by $\mathbb{CV}_k = \{CV_k^f \in \{0,1\} | f \in \mathbb{F}\}$, where $CV_k^f = 0$ means that content f is not cached and $CV_k^f = 1$ means that content f is cached. The CRV can receive immediate service if the content has already been cached. If not, the request will be sent to the RSU to which the CRV is connected. In the same way, the cache of RSU will be checked whether the requested content is cached. The cache state of each content for RSU n is indicated by $\mathbb{CR}_n = \{CR_n^f \in \{0,1\} | f \in \mathbb{F}\}$, where $CR_n^f = 0$ means that content f is not cached and $CR_n^f = 1$ means that content f is cached. If the content is located in the cache of RSU, it will be transmitted to the CRV with a certain communication cost. We call it *a cache hit* that the requested content could be available in the cache of CRV or RSU. Otherwise, the CRV has to fetch the desired content from the MBS (i.e., *a cache miss*), which will spend a higher communication cost. To sum up, the object of this caching framework is to provide content delivery services for smart vehicles with the communication cost as lower as possible.

Notably, the CRV may not be able to fetch all parts of the content in one time slot due to high mobility. Let $RE_k(t)$ denote the remaining size of the content requested by CRV k in the time slot t. When the CRV enters the coverage area of another RSU, the remaining part of the requested content will be transmitted subsequently.

3.2 Communication Model

In wireless communication, we consider that MBS and RSU allocate the orthogonal spectrum resources to CRVs such that there is no interference between wireless communications. The signal-to-noise ratio (SNR) between edge caching node i and CRV j at time slot t is given by

$$\gamma_{i,j}(t) = \frac{P_i g_{i,j}}{\sigma^2 + \sum_{v \in K \setminus \{j\}} P_i g_{i,v}}, \forall i \in \mathbb{N}, \forall j \in \mathbb{K} \tag{1}$$

where P_i is the transmission power of edge caching node i, $g_{i,j}$ is the channel gain between edge caching node i and CRV j, σ^2 is the power of additive white Gaussian noise.

Based on the assumption that the available spectrum resource is denoted as W^{mbs} Hz for MBS and W_n^{rsu} Hz for RSU n, $w_{i,j}(t)$ can be allocated as the continuous bandwidth resource to CRV j by edge caching node i. According to Shannon Theory, the data rate to fetch a segment of content f between edge caching node i and CRV j is given by [14]

$$r_{i,j}(t) = w_{i,j}(t) * \log_2 \left(1 + \gamma_{i,j}(t)\right), \forall i \in \mathbb{N}, \forall j \in \mathbb{K} \tag{2}$$

where $w_{i,j}(t)$ is the bandwidth resource allocated to CRV j and $\gamma_{i,j}(t)$ is the SNR at time slot t.

3.3 Request Model

The request of CRV k is denoted by $Q_k(t) = f \in \overline{\mathbb{F}}$, where $k \in \mathbb{K}, \overline{\mathbb{F}} = \mathbb{F} \cup \{0\}$. If there is no new request of CRV k, then $Q_k(t) = 0$. The vehicle will not submit a new request until the last request is served. We assume that the request history of one CRV follows a Markov chain and the next request only depends on the last request. We adopt the same vehicle request model as [17], where the request transition probability from content i to j of CRV k is given by

$$p_{i,j} = \begin{cases} P_0, & i \in \overline{\mathbb{F}}, j = 0 \\ (1 - P_0) \frac{\frac{1}{j^\beta}}{\sum_{j'=1}^{F} \frac{1}{j'^\beta}}, & i = 0, j \in \mathbb{F} \\ (1 - P_0) \frac{1}{H}, & i \in \mathbb{F}, j = (i + h) \bmod (F + 1), h \in \{1, 2, \ldots, H\} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

To be specific, P_0 indicates the probability that a vehicle does not have a new request in the current slot ($j = 0$). When the vehicle does not have a request in the last slot ($i = 0$), the request model depends on the content popularity, which follows a Zipf-like distribution, and β indicates the parameter of the distribution. Furthermore, each content $i \in \mathbb{F}$ has a set of H neighboring contents indicated as $\mathbb{H}_i = \{f \in \mathbb{F} : f = (i + h) \bmod (F + 1), \quad h \in \{1, 2, \ldots, H\}\}$, where H is the number of neighboring contents. The transition probability from content $i \in \mathbb{F}$ to its neighboring contents $j \in \mathbb{H}_i$ is modeled as a uniform distribution. In other words, the vehicle randomly selects a content from neighbor contents as the next request. Otherwise, the transition probability from $i \in \mathbb{F}$ to other contents $j \notin \mathbb{H}_i$ is zero.

For convenience of understanding, the notations used in this paper are summarized in Table 1.

Table 1. Summary of key notations.

Notation	Description
$\mathbb{N}$	index set of edge caching nodes(MBS, RSUs)
$\mathbb{K}$	index set of content requesting vehicles(CRVs)
$\mathbb{F}$	index set of all available contents
M_n	caching storage capacity of RSU $n \in \mathbb{N}$
L_k	caching storage capacity of CRV $k \in \mathbb{K}$
$CR_n^f(t)$	cache state of each content f for RSU n in time slot t
$CV_k^f(t)$	cache state of each content f for CRV k in time slot t
$RE_k(t)$	remaining size of the content requested by CRV k in time slot t
$r_{i,j}(t)$	data rate to fetch a segment of content f between cache node i and CRV j
$Q_k(t)$	request of CRV k at time slot t
$p_{i,j}$	request transition probability from content i to j of CRV k
$Loc_k(t)$	region index where CRV k is located at time slot t
$\varDelta CR_n^f(t)$	cache update action of RSU n for content f
$\varDelta CV_k^f(t)$	cache update action of CRV k for content f
$WM_{0,k}(t)$	allocated bandwidth to CRV k by MBS 0
$WR_{n,k}(t)$	allocated bandwidth to CRV k by RSU n
$c_{i,j}(t)$	transmission cost from edge caching node i to CRV j at time slot t

4 Problem Formulation

In this section, we formulate the joint optimization of content caching and delivery as a Markov decision process (MDP) problem. The detailed definitions of MDP, including state space, action space and reward, will be given.

4.1 State Space

At the beginning of each time slot, the agent will receive environment state information, including request content index, caching state, remaining size of content and vehicle locations. Specifically, the state space contains the following:

1. $Q_k(t) \in \overline{\mathbb{F}}$: request of CRV k at time slot t
2. $CR_n^f(t) \in \{0,1\}$: cache state of each content f for RSU n at time slot t
3. $CV_k^f(t) \in \{0,1\}$: cache state of each content f for CRV k at time slot t
4. $RE_k(t) \le S_{max}$: remaining size of the content requested by CRV k in time slot t, where S_{max} denotes the max size of all contents
5. $Loc_k(t) \in \{1,2,...,L\}$: location of CRV k at time slot t, which is represented by the region index the CRV is driving in. There will be more details about transport regions in Section 6.

The joint state space of the MDP process is denoted by $s(t) \in \mathbb{S}$:

$$s(t) = \{\mathbb{Q}(t), \mathbb{CR}(t), \mathbb{CV}(t), \mathbb{RE}(t), Loc(t)\}. \tag{4}$$

Thus, the size of the whole state space is $|\mathbb{S}| = (F+1)^K \times 2^{(N+K)F} \times S_{max}^K \times L^K$, which grows exponentially with the number of vehicles K and has poor scalability.

4.2 Action Space

After receiving the current environment state, the agent will decide which contents should be stored to which cache nodes and how to allocate the bandwidth resource, which would be done to the environment in order to achieve a lower communication cost. To be specific, the action space contains the following:

1. $\Delta CR_n^f(t) \in \{-1, 0, 1\}$: RSU cache update action, where -1 means deleting the content from the cache, 0 refers to maintaining the content cache states and 1 means inserting the content into the cache
2. $\Delta CV_k^f(t) \in \{-1, 0, 1\}$: CRV cache update action, which has the same meaning as $\Delta CR(t)$
3. $WM_{0,k}(t) \leq W^{mbs}$: allocated bandwidth to CRV k by MBS 0
4. $WR_{n,k}(t) \leq W_n^{rsu}$: allocated bandwidth to CRV k by RSU n

The joint action space of the MDP process is denoted by $a(t) \in \mathbb{A}$:

$$a(t) = \{\Delta\mathbb{CR}(t), \Delta\mathbb{CV}(t), \mathbb{WM}(t), \mathbb{WR}(t)\}. \tag{5}$$

Thus, the size of the whole action space is $|\mathbb{A}| = 3^{(N+K)F} \times (W^{mbs})^K \times (W_n^{rsu})^{NK}$. Same as above, the size of action space grows exponentially with the number of vehicles K.

4.3 Reward

The objective of this content caching and delivery policy is to provide content delivery services for smart vehicles with the communication cost as lower as possible. Based on this assumption, we design the following cost function to represent the transmission cost from edge caching node i(i.e., MBS and RSUs) to CRV j in the scenario of vehicle networks:

$$c_{i,j}(t) = p_i^b * \min\left(RE_j(t), r_{i,j}(t) * \Delta t\right), \forall i \in \mathbb{N}, \forall j \in \mathbb{K} \tag{6}$$

where p_i^b means the price per unit bandwidth for edge caching node i, which is higher for the MBS and lower for RSUs on account of different distances to CRVs. We assume that the service provider adopts resource-usage-based pricing [12]. If the remaining size of the content in one time slot (i.e., $RE_j(t)$) is smaller than the maximum amount of data transmitted in one time slot (i.e., $r_{i,j}(t) * \Delta t$), it will be billed according to the actual amount of data transmitted.

Further, the objective function can be given by the time-averaged transmission cost:

$$\min \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{i=0}^{N} \sum_{j=1}^{K} c_{i,j}(t). \tag{7}$$

Considering that general reinforcement learning algorithms are designed to maximize cumulative reward and we need to minimize the transmission cost, we apply the negative exponential function of the transmission cost as the reward function:

$$R(t) = e^{-\sum_{i=0}^{N} \sum_{j=1}^{K} c_{i,j}(t)}. \tag{8}$$

According to the Bellman equation, we can obtain the optimal policy for environment state s:

$$\mu^*(s) = \arg\max_{a \in \mathbb{A}} \left[R + \sum_{s\prime \in \mathbb{S}} \Pr\left(s' \mid s, a\right) V\left(s'\right) \right] \tag{9}$$

where s and a are the current state and action at time slot t, $s\prime$ is the next state at time slot $t+1$ and $V(*)$ is the value function of state.

5 Hierarchical Reinforcement Learning-based Caching and Delivery

In this section, we propose the hierarchical reinforcement learning (HRL)-based mobility-aware content caching and delivery policy for vehicle networks, in order to reduce the dimensions of state and action space and achieve better scalability and universality. In addition, we apply the LSTM algorithm to mine the mobility patterns of vehicles for the purpose of caching potential popular contents in advance.

According to the problem formulated in Section 4, the state space and action space are fairly massive and grow exponentially with the number of vehicles, which results in poor scalability. Convergence will take a long time if we simply apply traditional reinforcement learning algorithms, such as Q-learning and DQN. This leads to greatly reduced model scalability and actual application value. To address *the curse of dimensionality*, we apply the hierarchical reinforcement learning and divide the joint optimization of content caching and delivery into two subproblems (i.e., vehicle side and RSU side), as shown in Figure 2.

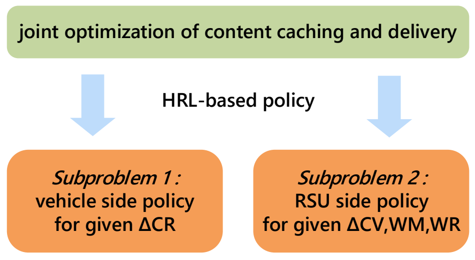

joint optimization of content caching and delivery
HRL-based policy
Subproblem 1 : vehicle side policy for given ΔCR
Subproblem 2 : RSU side policy for given ΔCV,WM,WR

Fig. 2. The framework of hierarchical reinforcement learning(HRL)-based policy.

Subproblem 1: vehicle side policy.

For given cache action in RSUs $\mathit{\Delta}\mathbb{CR}(t)$

$$\mu_V^*(s) = \underset{\mathit{\Delta}\mathbb{CV}(t),\mathbb{WM}(t),\mathbb{WR}(t)}{\operatorname{argmax}} \left[R + \sum_{s\prime \in \mathbb{S}} \operatorname{Pr}\left(s' \mid s, a\right) V\left(s'\right) \right]. \tag{10}$$

Subproblem 2: RSU side policy.

For given cache action in CRVs $\{\mathit{\Delta}\mathbb{CV}(t), \mathbb{WM}(t), \mathbb{WR}(t)\}$

$$\mu_R^*(s) = \underset{\mathit{\Delta}\mathbb{CR}(t)}{\operatorname{argmax}} \left[R + \sum_{s\prime \in \mathbb{S}} \operatorname{Pr}\left(s' \mid s, a\right) V\left(s'\right) \right]. \tag{11}$$

Note that the agent decision for cache action $\mathit{\Delta}\mathbb{CR}(t)$ in RSU side only affects the state $\mathbb{CR}(t)$, which has no influence on the vehicle side. In other words, adjusting the RSU side policy has much less influence on the vehicle side. Hence, we first optimize the vehicle side policy in subproblem 1 with RSU side policy fixed. We will apply DDPG algorithm to solve subproblem 1 in Section 5.1. Then, we will optimize the RSU side policy in subproblem 2 with vehicle side policy fixed and use Double-DQN algorithm to address subproblem 2 in Section 5.2.

5.1 Vehicle Side Policy

In this section, we fix the RSU side policy and optimize the vehicle side policy. In this case, the state space is simplified to $s(t) = \{\mathbb{Q}(t), \mathbb{CV}(t), \mathbb{RE}(t), Loc(t)\}$ and the action space is simplified to $a(t) = \{\mathit{\Delta}\mathbb{CV}(t), \mathbb{WM}(t), \mathbb{WR}(t)\}$. Accordingly, the size of state space decreases to $(F+1)^K \times 2^{KF} \times S_{max}^K \times L^K$ and the size of action space decreases to $3^{KF} \times (W^{mbs})^K \times (W_n^{rsu})^{NK}$. Considering the massive action space, we choose the Deep Deterministic Policy Gradient (DDPG) algorithm [6] to address this MDP problem. Different from the value-based DQN, DDPG is policy-based (i.e., directly output actions) and absorbs the advantages of DQN, which makes it more suitable for high-dimensional continuous action spaces. The architecture of DDPG is shown in Figure 3. In the following parts, we will discuss the detailed modules of DDPG.

Actor Network $\boldsymbol{\mu}$ Update. Different from stochastic policy, the actor network μ learns a deterministic policy $a = \mu(s|\theta^\mu)$ with the actor network parameter θ^μ. The input is the current state s and its output is the deterministic action $\mu(s|\theta^\mu)$, which is used to update the actor network parameter with the output of critic network $Q(s,a)$:

$$\nabla_{\theta^\mu}\mu = \nabla_a Q(s,a) \nabla_{\theta^\mu}\mu(s|\theta^\mu). \tag{12}$$

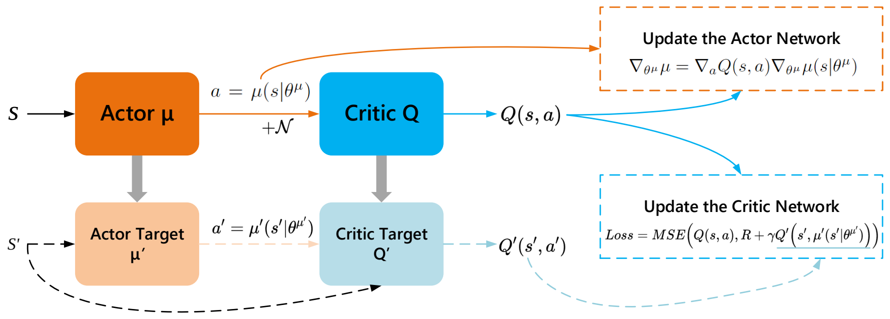

Update the Actor Network
$\nabla_{\theta^\mu}\mu = \nabla_a Q(s,a)\nabla_{\theta^\mu}\mu(s|\theta^\mu)$
S
Actor μ
$a = \mu(s|\theta^\mu)$
$+\mathcal{N}$
Critic Q
$Q(s,a)$
Update the Critic Network
$Loss = MSE\Big(Q(s,a), R+\gamma Q'\big(s',\mu'(s'|\theta^{\mu'})\big)\Big)$
S'
Actor Target μ'
$a' = \mu'(s'|\theta^{\mu'})$
Critic Target Q'
$Q'(s',a')$

Fig. 3. The architecture of DDPG.

Critic Network Q Update. The critic network Q is responsible for evaluating policies based on the action-value function $Q(s,a)$ with the critic network parameter θ^Q. The input is the current state s and the actual action $a_t = \mu(s|\theta^\mu) + \mathcal{N}$ where $\mathcal{N}$ denotes the Ornstein-Uhlenbeck noise [18] that functions as the exploration of policy. The output is the value function $Q(s,a)$ which is used to update the actor network parameter and calculate the TD error $y_t - Q(s,a)$ where y_t is the target value generated by critic target network Q'. The critic network parameter θ^Q will be updated by minimizing the loss:

$$\begin{aligned} Loss &= MSE(Q(s,a), y_t) \qquad (13)\\ &= MSE\left(Q(s,a), R+\gamma Q'\left(s', \mu'(s'|\theta^{\mu'})\right)\right). \qquad (14)\end{aligned}$$

Target Networks Update. The actor target network μ' and critic target network Q' are applied to calculate the target value y_t. Their architectures are consistent with the primary networks. However, they are updated slowly compared to the primary networks, which makes the learning performance stable and robust. Exponentially weighted moving average (EWAM) scheme is applied to update the target networks' parameters $\theta^{\mu'}$ and $\theta^{Q'}$:

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'} \qquad (15)$$

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'} \qquad (16)$$

where $\tau \in [0,1]$ is the weight parameter.

Experience Replay. DDPG draws on the experience replay of DQN. It constructs a replay memory to store a series of historical experiences $[s,a,c,s']$ to avoid sample-correlation during the training process. The network parameters can be updated by randomly choosing mini-batch samples from replay memory.

5.2 RSU Side Policy

In this section, we fix the vehicle side policy and optimize the RSU side policy. In this case, the state space is simplified to $s(t) = \{\mathbb{Q}(t), \mathbb{CR}(t), Loc(t)\}$ and the action space is simplified to $a(t) = \{\varDelta\mathbb{CR}(t)\}$, which are only related to RSUs. The dimension of state space decreases to $(F+1)^K \times 2^{NF} \times L^K$ and the dimension of action space decreases to 3^{NF}. As can be seen, the dimensions of state and action are reduced tremendously compared to the vehicle side and the action space is discrete. For the purpose of accelerating the convergence, we adopt the Double-DQN algorithm to solve subproblem 2.

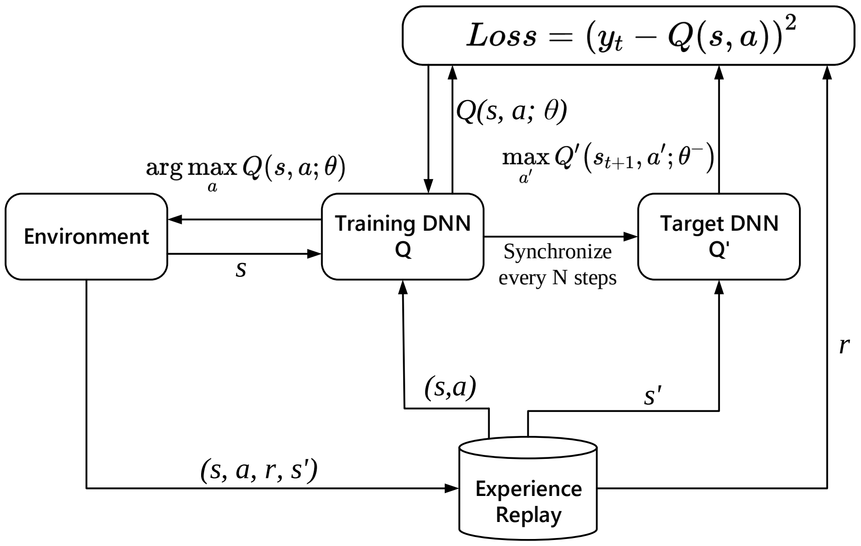

$Loss = (y_t - Q(s,a))^2$
$Q(s, a; \theta)$
$\max_{a'} Q'(s_{t+1}, a'; \theta^-)$
$\arg\max_a Q(s,a;\theta)$
Environment
Training DNN Q
Synchronize every N steps
Target DNN Q'
s
r
(s,a)
s'
(s, a, r, s')
Experience Replay

Fig. 4. The architecture of the DQN algorithm. The Double-DQN shares the same architecture and deep network as the DQN algorithm and differs from the DQN only in the calculation of the target value.

Double-DQN [20] is the variant of the DQN algorithm and has the same architecture and deep network as the DQN algorithm, as shown in Figure 4. In the DQN scheme, the deep neural network (DNN) Q is applied to estimate the state-action reward $Q(s,a)$. If we simply use one DNN, there are two key challenges in the training step: 1) The target is unstable, where the objective function for optimizing the DNN parameters depends on these parameters themselves; 2) The training samples are strongly correlated instead of independent, which makes the gradient descent towards a deterministic direction and there is a considerable probability that the training process will not converge. To address these challenges, DQN takes two measures accordingly: 1) Freezing target DNN Q': The parameters of target DNN Q' are fixed during several training steps until the parameters of DDN Q are synchronized to the target DNN Q', in order to keep the learning objective steady; 2) Experience replay memory D, which consists of interactions between the client and the environment. In each training step, DQN samples a batch of data from the experience memory as the training data and calculates the target value as follows:

$$y_t = \begin{cases} R_t & \text{if episode terminates at step t} + 1 \\ R_t + \gamma \max_{a'} Q'\left(s_{t+1}, a'; \theta^-\right) & \text{otherwise.} \end{cases} \tag{17}$$

The loss function is defined as the mean square error: $\left(y_t - Q(s,a)\right)^2$.

However, due to the max operation, the state-action value $Q(s,a)$ would be overestimated and the entire evaluation is overestimated accordingly since we always tend to select the action $\arg\max_{a'} Q'\left(s_{t+1}, a'; \theta^-\right)$. To avoid this upward bias, Double-DQN decouples the calculation operation by applying the training DNN Q to select actions, while employing the target DNN Q' to evaluate actions. The only difference between Double-DQN and DQN is the calculation of the target value, which alleviates the overestimation and instability:

$$y_t = \begin{cases} R_t & \text{if terminate at step t} + 1 \\ R_t + \gamma Q\left(s_{t+1}, \underset{a}{\operatorname{argmax}}\, Q'(s_{t+1}, a; \theta); \theta^-\right) & \text{otherwise.} \end{cases} \tag{18}$$

6 Experiments

In this section, we first introduce the experiment setup and the baseline schemes. Then the performance of our proposed scheme is evaluated on a real-world dataset and simulation results are given.

6.1 Experiment Setup

The schemes are implemented in Python 3.6 and experiments are run on a desktop computer with AMD R5 3600X CPU, 3.8GHz, 16G RAM under Ubuntu 18.04.5 LTS. Furthermore, we use the TensorFlow platform to implement the DDPG and Double-DQN algorithm of the content caching and delivery policy. The main parameters employed in the simulations are summarized in Table 2.

We apply the taxi trajectory dataset of the Xiamen island to simulate the mobility of vehicles, which consists of latitudes, longitudes and GPS times, etc. The road network of Xiamen island is used for the simulation, which contains 24,750 road nodes and 3,234 road segments. This road network covers the range of [118.0660E,118.1980E] $\times$ [24.4240N,24.5600N]. The Xiamen island is divided into 16 transport regions and a day is divided into 24 time slots.

Furthermore, we apply Long Short-Term Memory (LSTM) algorithm, which has proven to be an effective solution to time series prediction problems, to preprocess the trajectory data and mine vehicle mobility patterns before being

Table 2. Simulation Parameters.

Parameter	Value/Description
Number of RSUs	16
Number of CRVs	[30, 60]
Number of contents	[10, 40]
Size of contents	[20, 80]MB
Storage Capacity of RSU	200MB
Storage Capacity of CRV	[80, 120]MB
Bandwidth of MBS, RSUs	[5, 20]MHz
Transmission Power of MBS	35dBm
Transmission Power of RSU	33dBm
Power of Gaussian Noise	-95dBm

fed to the RL agent. Specifically, we apply 24 units of LSTM to predict the location in the next time slot. At the beginning of each time slot, the vector of the previous latitudes and longitudes would be given to the input of LSTM. The output of LSTM is the latitude and longitude where vehicles may be in this time slot. Then the latitude and longitude will be transferred into the index of transport region, which is served as the location of CRVs at time slot t (i.e., $Loc_k(t)$ in Section 4.1).

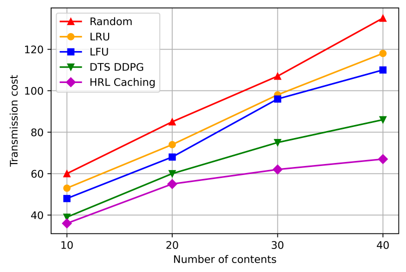
6.2 Baseline Schemes

For performance comparison, we present the baseline schemes as follows:

- *Random*: The contents cached in RSUs and CRVs are randomly selected from all of the available contents.
- *Least Recently Used (LRU)*: When the cache capacity of RSU or CRV is already full, the least recently used content will be evicted. In other words, the longest unrequested content will be removed.
- *Least Frequently Used (LFU)*: When the cache capacity of RSU or CRV is already full, the least frequently used content will be evicted. In other words, the content with the smallest request frequency will be removed.
- *Double Time-Scale DDPG (DTS DDPG)* [12]: The cooperative caching problem is modeled as a double time-scale Markov decision process (DTS-MDP). The content caching decision is made on the large time-scale while the joint decision of vehicle scheduling and bandwidth allocation is implemented on the small time-scale. The DDPG algorithm is implemented to obtain a sub-optimal solution.

6.3 Simulation Results

Impact of the Number of Contents. As shown in Figure 5, we compare the transmission cost for different numbers of contents. We vary the number of contents from 10 to 40. The capacity of CRVs is 100MB and the number of CRVs



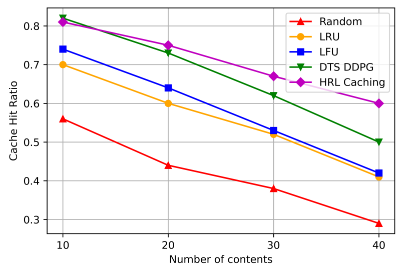
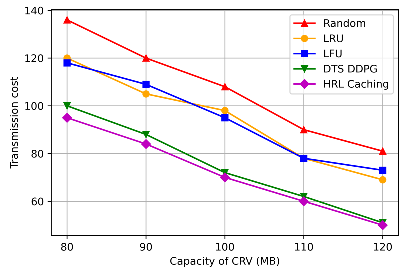


Fig. 5. Impact of the number of contents on the transmission cost.

Fig. 6. Impact of the number of contents on the cache hit ratio.

is 50. It can be observed that the transmission costs of five approaches all obviously increase with the increase of the number of contents. This is because the more contents there are, the more cache replacement is needed, which causes an increase in the transmission cost. As expected, the proposed HRL-based caching policy performs better with various numbers of contents compared to other baseline schemes. The line chart in Figure 6 demonstrates that the cache hit ratio has a decreasing trend for all caching schemes as the number of contents increases. Our proposed approach still performs better than other baseline schemes and the advantage is even greater when the number of contents increases.



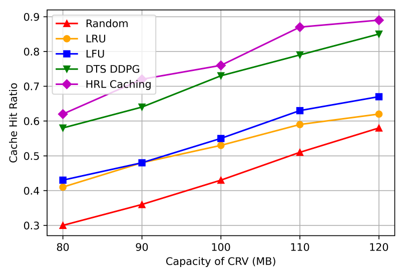


Fig. 7. Impact of the capacity of CRVs on the transmission cost.

Fig. 8. Impact of the capacity of CRVs on the cache hit ratio.

Impact of the Capacity of CRVs. Then, we compare the transmission cost and cache hit ratio for different cache capacities of CRVs. We change the cache capacity of CRVs from 80MB to 120MB. As we can see in Figure 7, the increasing cache capacity has a positive impact on the transmission cost. The transmission cost decreases with the increase of cache capacity, especially for our proposed HRL-based caching policy. This is reasonable because a larger cache capacity enables CRVs to cache more popular contents simultaneously, which will reduce the number of wireless communications to the MBS or RSUs. In Figure 8, as the cache capacity grows, our proposed approach is the best scheme and the Random scheme has the worst performance. This is because the Random scheme does not consider the content popularity and has no ability to predict the next request.

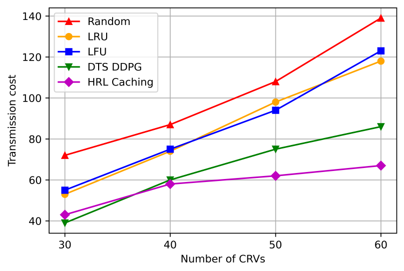


Fig. 9. Impact of the number of CRVs on the transmission cost.

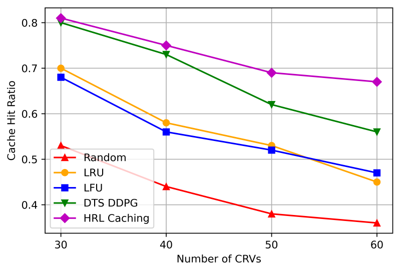


Fig. 10. Impact of the number of CRVs on the cache hit ratio.

Impact of the Number of CRVs. In addition, we change the number of CRVs to compare the transmission cost and the cache hit ratio. We change the number of CRVs from 30 to 60 and the number of contents is set to 100. The capacity of CRVs is set to 100MB. Figure 9 illustrates that our approach achieves the best performance and the gap between HRL-based caching policy and DTS DDPG policy becomes larger with the increasing number of CRVs. This reveals that our approach has better scalability than DTS DDPG and is more suitable for large-scale vehicle network scenarios. In Figure 10, the cache hit ratio decreases with the increasing number of CRVs. This is because the cache capacity is limited and unable to satisfy all requests of vehicles. Furthermore, it is worth noting that the LRU and LFU schemes have similar performances, as they are both driven by simple statistics of request history and mine user request patterns to guide the cache decision.

The Convergence Performance. Figure 11 shows the convergence performance of our proposed HRL approach and other baseline schemes. The number of contents is fixed as 100. The capacity of CRVs is set to 100MB and the number of CRVs is 50. It can be observed that for HRL-based caching policy and DTS DDPG policy, the total content transmission cost of each episode decreases rapidly and gradually maintains a relatively stable value with the increase of training episodes. Meanwhile, there are no significant changes occurring in the transmission cost with the increase of episodes for Random, LRU and LFU schemes. This is consistent with the fact that they are not reinforcement

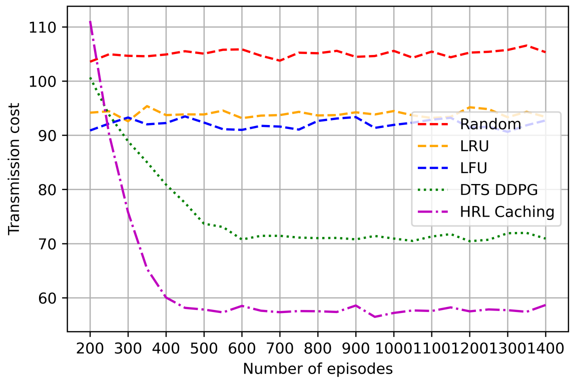


Fig. 11. The convergence performance.

learning-based schemes. In addition, the HRL-based caching policy converges at about 400 episodes, which is obviously faster than DTS DDPG. This shows that it has better scalability from another aspect.

7 Conclusion

In this paper, we focus on the main challenges in vehicle networks, including high mobility and scalability. An HRL-based mobility-aware content caching and delivery policy for vehicle networks is proposed to achieve better scalability and generality. The joint optimization of content caching and delivery is decomposed into two subproblems. DDPG and Double-DQN are adopted to deal with sequential decision problems. Experimental results demonstrate that our approach reduces the dimension of state space and outperforms other baseline schemes in terms of transmission cost and convergence speed. Our approach still has huge room for improvement. In the future, we will continue to improve our scheme in terms of more accurate location prediction and collaboration between edge caching nodes.

References

1. Abbas, N., Zhang, Y., Taherkordi, A., Skeie, T.: Mobile edge computing: A survey. IEEE Internet of Things Journal **5**(1), 450–465 (2018). https://doi.org/10.1109/JIOT.2017.2750180
2. Barto, A.G., Mahadevan, S.: Recent advances in hierarchical reinforcement learning. Discrete Event Dynamic Systems **13**(1), 41–77 (2003). https://doi.org/10.1023/A:1022140919877
3. Hu, B., Fang, L., Cheng, X., Yang, L.: Vehicle-to-vehicle distributed storage in vehicular networks. In: 2018 IEEE International Conference on Communications (ICC). pp. 1–6 (2018). https://doi.org/10.1109/ICC.2018.8422220

4. Hu, Y.C., Patel, M., Sabella, D., Sprecher, N., Young, V.: Mobile edge computing - a key technology towards 5g. ETSI white paper **11**(11), 1–16 (2015)
5. Jiang, W., Feng, G., Qin, S.: Optimal cooperative content caching and delivery policy for heterogeneous cellular networks. IEEE Transactions on Mobile Computing **16**(5), 1382–1393 (2017). https://doi.org/10.1109/TMC.2016.2597851
6. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. In: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings (2016), http://arxiv.org/abs/1509.02971
7. Majidi, F., Khayyambashi, M.R., Barekatain, B.: Hfdrl: An intelligent dynamic cooperate cashing method based on hierarchical federated deep reinforcement learning in edge-enabled iot. IEEE Internet of Things Journal **9**(2), 1402–1413 (2022). https://doi.org/10.1109/JIOT.2021.3086623
8. Narayanan, A., Verma, S., Ramadan, E., Babaie, P., Zhang, Z.: Deepcache: A deep learning based framework for content caching. In: Proceedings of the 2018 Workshop on Network Meets AI ML. pp. 48–53 (2018). https://doi.org/10.1145/3229543.3229555
9. Nomikos, N., Zoupanos, S., Charalambous, T., Krikidis, I.: A survey on reinforcement learning-aided caching in heterogeneous mobile edge networks. IEEE Access **10**, 4380–4413 (2022). https://doi.org/10.1109/ACCESS.2022.3140719
10. Pateria, S., Subagdja, B., Tan, A.h., Quek, C.: Hierarchical reinforcement learning: A comprehensive survey. ACM Comput. Surv. **54**(5) (jun 2021). https://doi.org/10.1145/3453160
11. Qian, Y., Wang, R., Wu, J., Tan, B., Ren, H.: Reinforcement learning-based optimal computing and caching in mobile edge network. IEEE Journal on Selected Areas in Communications **38**(10), 2343–2355 (2020). https://doi.org/10.1109/JSAC.2020.3000396
12. Qiao, G., Leng, S., Maharjan, S., Zhang, Y., Ansari, N.: Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks. IEEE Internet of Things Journal **7**(1), 247–257 (2020). https://doi.org/10.1109/JIOT.2019.2945640
13. Qin, Z., Xian, Y., Zhang, D.: A neural networks based caching scheme for mobile edge networks: Poster abstract. In: Proceedings of the 17th Conference on Embedded Networked Sensor Systems. p. 408409 (2019). https://doi.org/10.1145/3356250.3361961
14. Rappaport, T.S., et al.: Wireless communications: principles and practice, vol. 2. prentice hall PTR New Jersey (1996)
15. Song, C., Xu, W., Wu, T., Yu, S., Zeng, P., Zhang, N.: Qoe-driven edge caching in vehicle networks based on deep reinforcement learning. IEEE Transactions on Vehicular Technology **70**(6), 5286–5295 (2021). https://doi.org/10.1109/TVT.2021.3077072
16. Sun, Y., Chen, Z., Liu, H.: Delay analysis and optimization in cache-enabled multi-cell cooperative networks. In: 2016 IEEE Global Communications Conference (GLOBECOM). pp. 1–7 (2016). https://doi.org/10.1109/GLOCOM.2016.7841723
17. Sun, Y., Cui, Y., Liu, H.: Joint pushing and caching for bandwidth utilization maximization in wireless networks. IEEE Transactions on Communications **67**(1), 391–404 (2019). https://doi.org/10.1109/TCOMM.2018.2858791
18. Szepesvári, C.: Algorithms for reinforcement learning. Synthesis lectures on artificial intelligence and machine learning **4**(1), 1–103 (2010). https://doi.org/10.2200/S00268ED1V01Y201005AIM009

19. Tsai, K.C., Wang, L., Han, Z.: Mobile social media networks caching with convolutional neural network. In: 2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW). pp. 83–88 (2018). https://doi.org/10.1109/WCNCW.2018.8368988
20. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 30 (2016). https://doi.org/10.1609/aaai.v30i1.10295
21. Yao, J., Ansari, N.: Joint content placement and storage allocation in c-rans for iot sensing service. IEEE Internet of Things Journal **6**(1), 1060–1067 (2019). https://doi.org/10.1109/JIOT.2018.2866947
22. Yu, Z., Hu, J., Min, G., Zhao, Z., Miao, W., Hossain, M.S.: Mobility-aware proactive edge caching for connected vehicles using federated learning. IEEE Transactions on Intelligent Transportation Systems **22**(8), 5341–5351 (2021). https://doi.org/10.1109/TITS.2020.3017474
23. Zong, T., Li, C., Lei, Y., Li, G., Cao, H., Liu, Y.: Cocktail edge caching: Ride dynamic trends of content popularity with ensemble learning. In: IEEE INFOCOM 2021 - IEEE Conference on Computer Communications. pp. 1–10 (2021). https://doi.org/10.1109/INFOCOM42981.2021.9488910