Multi-Agent Reinforcement Learning for Cooperative Edge Caching in Internet of Vehicles

Kai Jiang*, Huan Zhou*, Deze Zeng†, and Jie Wu‡
*College of Computer and Information Technology, China Three Gorges University, Yichang, China
†School of Computer Science, China University of Geosciences, Wuhan, China
‡ Department of Computer and Information Sciences, Temple University, Philadelphia, USA
jiangkai0112@gmail.com, zhouhuan117@gmail.com, deze@cug.edu.cn, jiewu@temple.edu

***Abstract*—Edge caching has been emerged as a promising solution to alleviate the redundant traffic and the content access latency in the future Internet of Vehicles (IoVs). Several Reinforcement Learning (RL) based edge caching methods have been proposed to improve the cache utilization and reduce the backhaul traffic load. However, they can only obtain the local sub-optimal solution, as they neglect the influence of environment by other agents. In this paper, we investigate the edge caching strategy with consideration of the content delivery and cache replacement by exploiting the distributed Multi-Agent Reinforcement Learning (MARL). We first propose a hierarchical edge caching architecture for IoVs and formulate the corresponding problem with the objective to minimize the long-term cost of content delivery in the system. Then, we extend the Markov Decision Process (MDP) in the single agent RL to the multi-agent system, and propose a distributed MARL based edge caching algorithm to tackle the optimization problem. Finally, extensive simulations are conducted to evaluate the performance of the proposed distributed MARL based edge caching method. The simulation results show that the proposed MARL based edge caching method significantly outperforms other benchmark methods in terms of the total content access cost, edge hit rate and average delay. Especially, our proposed method greatly reduces an average of 32% total content access cost compared with the conventional RL based edge caching methods.**

***Index Terms*—edge caching; multi-agent reinforcement learning; content delivery; cache replacement; markov decision process**

I. Introduction

Along with the recent advances in wireless access technologies and the steady growing of vehicles on the roads, the traditional technology-driven transportation system is evolving from an era of providing simple transportation services into a more powerful data-driven intelligent era. As an important paradigm in the Fifth Generation (5G) networks, Internet of Vehicles (IoVs) enable vehicles to provide reliable vehicular multimedia services, cooperative cruise control and traffic-related information management, which have paved a path towards intelligent transportation, as well as enhanced driving safety and travel comfort for drivers and passengers [1], [2]. In the meanwhile, the flourishing vehicular applications require vehicles to access huge amount of Internet data, especially for some delay-sensitive contents (e.g., video, image-aided navigation and live traffic information), leads to significant redundant traffic loads and considerable delay for content delivery in the network. Nevertheless, due to the centralized characteristic of current cellular networks architectures, long transmission distance and the limited channel bandwidth all pose unprecedented challenges for supporting massive content delivery while also satisfying the tight Quality of Service (QoS) requirement in the future IoVs [3], [4], [5].

Indeed, the rapid increase in cost and latency driven by the transmission distance in cloud-based processing has become a severe issue in IoVs and attracted widespread academic concern recently [6], [7]. Despite continuous efforts have been spent on enhancing the channel bandwidth by adopting sophisticated techniques, the utilization efficiency of the radio spectrum is notably reaching its theoretical boundary [8]. Thus, such efforts will be not sufficient and a fundamental innovation that breaks the bottleneck of massive content delivery in IoVs is urgently required.

Fortunately, stemming from the recent studies, different popular contents require different levels of priority. Explicitly, only a few popular contents are repeatedly downloaded upon request from the majority of vehicles, whilst the remaining large portion of the contents impose rather infrequent access demands [9], [10]. This pattern has promoted the implementation of edge caching technique in IoVs. Specifically, edge caching is emerged as a promising solution to alleviate the redundant traffic and lower the content access latency in the future IoVs, which caches popular contents in close proximity to vehicles by utilizing the storage resources at intermediate Roadside Units (RSUs). Hence, vehicles are able to access popular contents from the caching-enabled nearby RSUs directly, instead of downloading them from the remote server via backhaul networks [11].

Extensive studies have focused on edge caching strategies in IoVs [12]-[16]. However, these aforementioned typical edge caching strategies cannot be applied to the dynamic IoVs directly, due to the limited storage capacities of RSU in each cell and the time-varying popularity of contents as well as the tight constraint on content delivery deadline. In addition, the neighboring RSUs are enabled to communicate and share popular contents with each other, the synergies between their cache should also be harnessed to achieve good performance. Therefore, to improve the cache utilization in all cooperative RSUs and reduce the backhaul traffic load, it is crucial that an efficient edge caching strategy with content delivery and cache replacement is available in the IoVs.

Recently, with the revival of Artificial Intelligence (AI), emerging Reinforcement Learning (RL) has exhibited great potential for efficient edge caching in IoVs. Without any prior knowledge, RL can interact with the dynamic environment to enhance the intelligence of IoVs, and thus learn the optimal edge caching strategies. Specifically, the authors in [17] utilized RL algorithm to investigate the flexible cache decision-making strategies, so as to minimize the aggregate cost across contents and time. In [18], a RL based method was also exploited for dynamically orchestrate edge computing and caching resources to improve system utility. In [19], a RL based self-adaptive edge caching framework is proposed, with the aim to reduce the access delay of users as well as the traffic burden on the backhual. To jointly optimize the content placement and content delivery in IoVs, a deep RL based method is proposed to minimize the content access cost in [20].

In general, most studies about RL based edge caching problems neglected the influence of environment by other agents when a certain particular agent interacts and learns the environment independently. Therefore, only the local sub-optimal solution can be obtained in the system, not global optimal solution, especially considering a long term optimization [21]. Besides, conventional centralized RL is easy to be trapped in the curse of dimensionality by the exponential increase of state-action space in practical dynamic scenarios [22], [23]. In this paper, we investigate the edge caching strategy with consideration of the content delivery and cache replacement by exploiting the distributed Multi-Agent Reinforcement Learning (MARL). We first propose a hierarchical edge caching architecture for IoVs, where the cooperative caching among multi-RSUs and Macro Base Station (MBS) is utilized to minimize the content access cost and the backhaul traffic in the system. Furthermore, we extend the Markov Decision Process (MDP) in the single-agent RL to the multi-agent system and tackle this optimization problem with a distributed MARL based edge caching algorithm. The key contributions can be summarized as follows:

1) We present the hierarchical edge caching architecture for IoVs and formulate the corresponding problem with the objective to minimize the long-term overhead of content delivery in the system.
2) To address the overhead minimization problem, we first introduce a MDP for the optimization of cache replacement process in an available RSU.
3) Then, we extend the MDP to a multi-agent system, and further tackle the optimization problem with a MARL based edge caching method.
4) Simulation results demonstrate that our proposed method greatly outperforms other caching strategies in different scenarios.

The remainder of this paper is organized as follows. Section II describes the system framework followed by the process of content delivery and cache replacement. Section III formulates the problem with the objective to minimize the long-term cost of content delivery in the system. Section IV introduces our MARL based edge caching method and its design in cooperative edge caching. Furthermore, we analyze the simulation results in section V. Conclusion and future work are introduced in section VI.

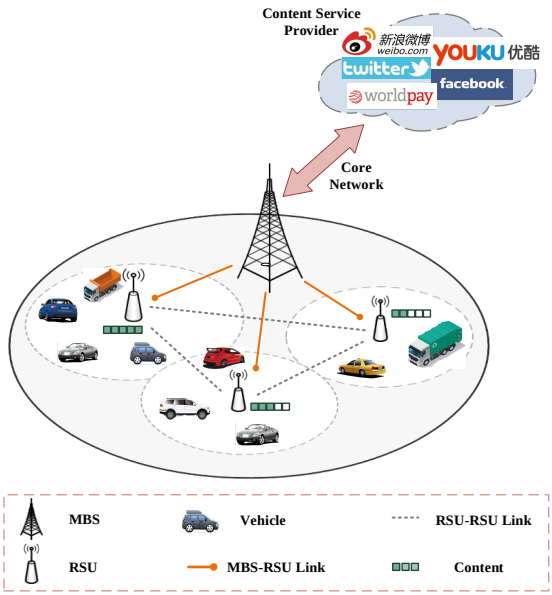


Fig. 1. System Model.

II. AI-EMPOWERED VEHICULAR EDGE NETWORK ARCHITECTURE

In this section, we first present the network architecture adopted in this paper, and then introduce the process of content delivery and cache replacement in datails.

A. Network Infrastructure

We consider a cooperative edge caching-supported IoV architecture of one macro-cell with an MBS and N small cells, each of which with a RSU, as shown in Fig. 1. The set of these RSUs is denoted by $\mathcal{N} = \{A_1, A_2, \ldots, A_N\}$. Considering the sufficient cache capacity, MBS is assumed to be connected with the Service Providers (SPs) via the core link and able to cache all available popular contents. Here, we let $\mathcal{F} = \{1, 2, \ldots, F\}$ denote the index set of F available contents that are supported by the SPs, and s_f denote the size of content $f (f \in \mathcal{F})$. In addition, each RSU is endowed with limited available cache capacity of R, such that some popular contents can be cached in the RSUs to reduce the content delivery cost. All RSUs can be interoperable with each other and connected to the MBS via wired line. I vehicles (denoted as $\mathcal{U} = \{1, 2, \ldots, I\}$) randomly distribute in the wireless coverage scope and request various popular contents frequently via cellular links for a relatively long period. We assume that those vehicles are located in at least one small cell covered by a RSU. Furthermore, we assume that the system operates in fixed length of time slots $\mathcal{T} = \{0, 1, 2, \ldots, t\}$, where t denotes the finite time horizon. In each time slot, a

vehicle can request only one content, while the caching and delivery decisions of each RSU are also periodically updated.

Vehicles may have different preferences for popular contents, we use content popularity $\rho_f(f \in \mathcal{F})$ to reflect the probability distribution of all requested contents in the content library, i.e. ρ_f represents the probability of requesting the content f among all the content requests by various vehicles. Similar to the studies [12] [24], we assume that the content popularity follows the Mandelbrot-Zipf distribution. Hence, the expected probability of request for content f can be expressed as:

$$\rho_f = \frac{R_f^{-\theta}}{\sum_{i \in \mathcal{F}} R_i^{-\theta}}, \quad \forall f \in \mathcal{F} \tag{1}$$

where R_f is the rank of content f in the descending order of content popularity; $\theta > 0$ is the skewness factor, and the content popularity will become more concentrated when θ becomes larger.

Within the coverage of MBS, each RSU can cache various contents to satisfy the demands of content services for vehicles. Caching contents in RSUs allows the requested contents much closer to vehicles, instead of excessively downloading duplicated contents from remote MBS. Particularly, when a vehicle sends a specific content request to the local RSU within the range of the small cell, the local RSU will search its cache space first to make sure whether the desired content is cached in itself. If the requested content is not found in its cache, the local RSU can obtain it either from the adjacent RSUs or download it directly from the MBS through backhaul links, and then deliver it to the vehicle. Meanwhile, all cooperative RSUs may replace its cache with the popular contents based on the vehicles' requests in each time slot.

B. Content Delivery Model

In this paper, we use T_1, T_2, T_3 to denote the content transmission delay among MBS, cooperative RSUs and vehicles, respectively. Specifically, T_1 denotes the transmission delay from a requested vehicle to its local RSUs, which is equal to the value of content size divided by the wireless downlink data rate between local RSU and the vehicle; T_2 denotes the transmission delay through RSU-RSU link; T_3 denotes the transmission delay through RSU-MBS link. Note that we neglect the transmission delay of sending request identifier by the vehicle due to its tiny data size and the high link rate in most instance. Here, the wireless downlink rate can be derived by

$$r_{n,i} = B_n \log_2 \left(1 + \frac{P_i g_{i,n}}{\sigma^2 + \sum_{v \in \mathcal{U} \setminus \{i\}} P_v g_{v,n}}\right) \tag{2}$$

where B_n denotes the channel bandwidth, P_i is the transmission power, σ^2 represents the power of intricate white Gaussian noise, and $g_{i,n}$ is the channel gain of wireless propagation channel. Without loss of generality, the interference management is also considered for this wireless communication scenario.

Naturally, caching nodes (cooperative RSUs or MBS) will incur certain costs for delivering the requested contents to the vehicles. Here we use C_1, C_2, and C_3 to denote the incurred transmission cost through Vehicle-RSU, RSU-RSU, and MBS-RSU links, respectively, shown in Fig. 2. Specifically, the incurred transmission cost through different links can be calculated as the product of the link bandwidth times the content delivery delay and the price of unit leased communication resource of the corresponding link. We assume that the communications of MBS-RSU and RSU-RSU are all via wired optical cables.

The decision of any local RSU $i(i \in \mathcal{N})$ for the requested content f can be represented by the binary decision variable $a_{f,i,j}^t \in \{0, 1\}$ in time slot t. When a content request occurs, the aggregate overhead through different links can be analyzed in detail as follows:

- $a_{f,i,i}^t$ = 1 means that the content f is cached in the local RSU i at time slot t, and it can be delivered to the vehicle directly. In this case, the total overhead $C_{i,i}$ is just the transmission cost C_1 from the local RSU to the vehicle;
- $a_{f,i,j}^t$ = 1 means that the content f is not cached in the local RSU i, but at least one of cooperative RSUs have cached the desired content, and then the local RSU can inquire and obtain it from the cooperative RSU j for the vehicle. Therefore, the total overhead $C_{i,j}$ consists of the transmission cost C_2 from the RSU j to the local RSU i and the transmission cost C_1;
- $a_{f,i,N+1}^t$ = 1 means that there is no desired content in the local RSU i and cooperative RSUs, and then local RSU i will forward the request identifier to MBS for processing, i.e. local RSU i can download the content f from MBS directly in time slot t. In this way, the total overhead $C_{i,N+1}$ consists of the transmission cost C_3 from MBS to the local RSU i and the transmission cost C_1.

C. Cache Replacement Model

Considering the practical dynamic property of content request in IoVs, a wise content replacement strategy is indispensable to efficiently manage the cache space. In the cache replacement phase, each cooperative RSU is able to update its cached contents to improve the overall cache utilization and effectiveness, as well as reduce the long-term overhead of content delivery in the system.

Once the new requested contents come from other adjacent RSUs or MBS in a time slot, the local RSU is supported to determine whether these contents need to be cached, and if needed, some contents in its existing cache may be replaced due to the finite cache capacity. Therefore, the problem is whether and which content should be replaced with the new ones when the cache capacity is fully occupied? In this paper, the content replacement control in RSU i can be represented by $\boldsymbol{c}_{f,i,i}^t = \left[c_{1,i,i}^t, c_{2,i,i}^t, \ldots, c_{F,i,i}^t\right]$ in time slot t, where $c_{f,i,i}^t \in \{0, 1\}$ $(f \in \mathcal{F})$ indicates whether and which content in RSU i should be replaced by the current content. Furthermore, in order to utilize the cooperative multipoint caching between RSUs efficiently, we will introducte a utility

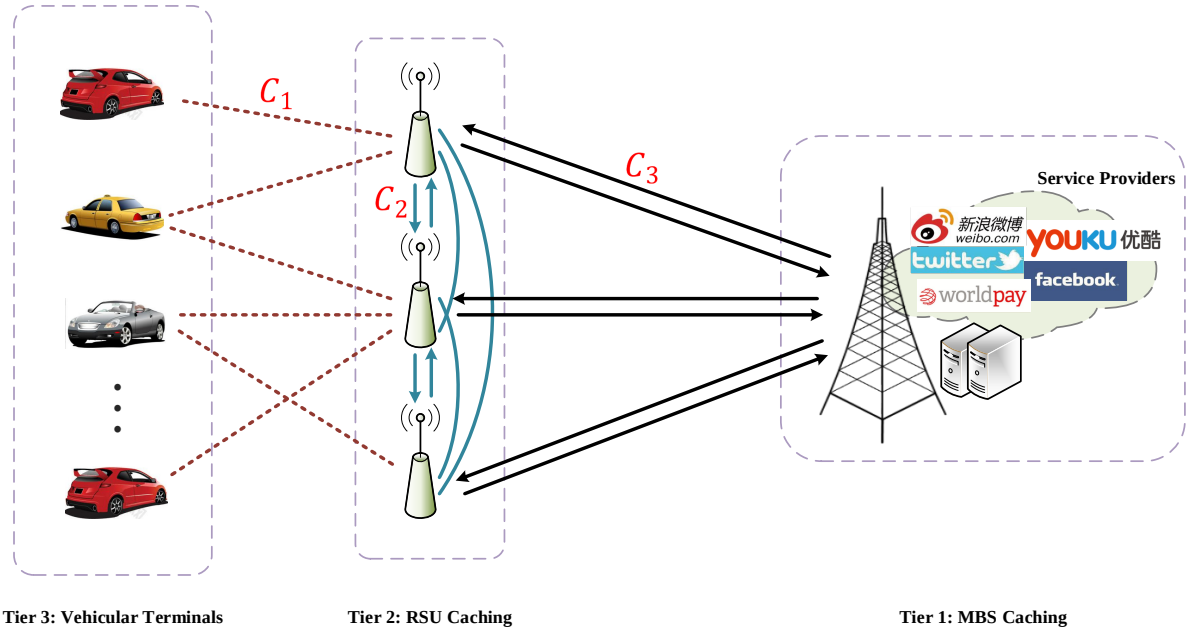


Fig. 2. Content Delivery Process in Hierarchical Networks.

based cache replacement strategy in subsequent sections. The utility value can be calculated based on the content delivery cost incorporating different factors.

III. Problem Formulation

A. Objective Function

The RSUs can cooperatively provide distributed edge caching to make full use of the available cache resources. This paper aims to minimize the long-term overhead of content delivery in the system while satisfying some specific constraints, and the objective function of the optimal cache replacement is formulated as follows:

$$\begin{aligned} &\min_{a_{f,i,j}} \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{f=1}^{F} \sum_{i=1}^{N} \sum_{j=1}^{N+1} \rho_f a_{f,i,j}^t C_{i,j} \\ s.\ t.\quad & C1: a_{f,i,j}^t \in \{0,1\}, \quad \forall f, \forall i, \forall j \\ & C2: a_{f,i,j}^t \le a_{f,j,j}^t, \quad \forall f, \forall i, \forall j \\ & C3: a_{f,N+1,N+1}^t = 1, \quad \forall f \\ & C4: \sum_{j=1}^{N+1} a_{f,i,j}^t = 1, \quad \forall f, \forall i \\ & C5: \sum_{f=1}^{\mathcal{F}} a_{f,i,i}^t \cdot s_f \le R_i, \quad \forall i. \end{aligned} \tag{3}$$

Here, C1 is the integer nature of the binary caching decision; C2 guarantees that the content request can be answered by the cooperative RSUs or MBS which have cached the requested content; C3 denotes that the MBS has cached all available popular contents by its sufficient cache capacity; C4 ensures that the content requested by a certain vehicle can only be handled by a RSU or MBS ultimately at one time slot; C5 is the limitation of RSU's cache capacity, which guarantees that the total data size of the cached contents on a RSU should not exceed its cache capacity.

B. Problem and Challenges

To address the above problem in (3), it is necessary to find the optimal results of caching decision variables $\left\{a_{f,1,j}^t, a_{f,2,j}^t, \cdots, a_{f,N,j}^t\right\}$ in the current time slot, which can be used to minimize the overall overhead of content delivery in the system with the given constraints. However, the caching decision variable $a_{f,i,j}^t$ of any RSU is a binary variable and dynamically changing. The system needs to collect a large amount of network state information, and makes the global decision on cache replacement and content delivery for each RSU based on the network's current state. Moreover, we devote to a more practical case in which the prior information of content request pattern within a time period is unknown in advance. Therefore, the objective function is NP-hard undoubtedly. The feasible set of the problem is not convex and the complexity is very enormous, so conventional methods may not adapt to the dynamic property and make intelligent caching decisions.

C. Markov Decision Process

We approximate the optimization of cache replacement process in an available RSU as a Markov Decision Process (MDP) to minimize the overhead of content delivery.

Three critical elements are identified in MDP as follows:

- ***State Space*** **:** The state in RL is a space to reflect the IoVs environment. Accordingly, in our proposed system, the current state of an available RSU i can be given

as $\boldsymbol{z_t} = \left(\boldsymbol{a^t_{f,i,i}}, \boldsymbol{q^t_{i,r}}\right)$. As described earlier, the former $\boldsymbol{a^t_{f,i,i}} = \left(a^t_{1,i,i}, a^t_{2,i,i}, \ldots, a^t_{F,i,i}\right)$ denotes the current cache status respect to the content $f \in \mathcal{F}$ in RSU i. The latter $\boldsymbol{q^t_{i,r}}$ is the arrived content requests in the current time slot t. Above information will be assembled as a state and sent to the agent in each slot.

- ***Action Space*** **:** The objective of an agent is to map the space of states to the space of actions. In this system, the action consists of two parts, $\boldsymbol{a^t_{f,i,i}}$ and $\boldsymbol{c^t_{f,i,i}}$, where matrix $\boldsymbol{a^t_{f,i,i}}_{\boldsymbol{F\times N+1}}$ indicates the decision of RSU i for the current requested content $f \in \mathcal{F}$, and the vector $\boldsymbol{c^t_{f,i,i}}$ is the content replacement control in RSU i in slot t.
- ***Reward Function*** **:** In general, the immediate network reward function is related to the objective function. In the considered optimization problem, our objective is to obtain the minimum long-term overhead of content delivery in the system, and the goal of RL is to achieve the maximum reward. Thus, the value of reward needs to be negatively correlated to the value of the overhead. To minimize the long-term overhead of content delivery in system, we define the immediate reward as normalized $\mathcal{R}(\boldsymbol{z_i(t)}, \boldsymbol{d_i(t)}) = e^{-\sum_f^{\mathcal{F}} \sum_j^{N+1} \rho_f a^t_{f,i,j} C_{i,j}}$, where negative exponential function is used to transform the problem successfully.

IV. MARL-Empowered Vehicular Edge Caching

Most programming problems which are controlled by decision making in dynamic system can be described in terms of MDP. In general, MDP can be solved by linear programming or dynamic programming algorithms. But when the transition probability and the immediate reward are time-varying and unknown in discrete time steps, the RL based methods are more suitable to deal with these problems. There are many mature algorithms in RL. Thereinto, Q-learning is widely used because of its straightforward structure, fast convergence speed and ease of use. In this section, we briefly introduce the classical Q-learning algorithm, and then extend Q-learning to a multi-agent system to optimize the caching strategy under the stochastic game framework.

A. Reinforcement Learning : Q-Learning

Q-learning is an effective model-free RL approach based on value iteration, for which both environment and the state transition probability are not explicit [25]. Q-learning enables the agent to automatically learn the optimal behaviour separately within a specific context in each time step. The agent of Q-learning needs to compute the result of Q-function of each state-action pair, and updates the Q value in a maintained dimension Q-table after each interaction. Specifically, the agent observes current state of the environment as z_t, then selects and executes an action d_t from its admissible action space D according to a policy π in each time step. After that, the agent will transfer into the next new state z_{t+1} with the transition probability $p_{z_t z_{t+1}}(d_t)$, and obtain an immediate reward $r_t = r(z_t, d_t)$. The goal of the agent is to find an optimal policy $\pi^*(z_t) = d_t^* \in D$, so that it can obtain the maximum cumulative discounted reward $R^t_{max} = \sum_0^\infty \gamma^t r_t$, where $\gamma \in (0,1)$ is the discounting factor indicating the importance of the predicted future rewards. The state value function $V^\pi(z_t)$ is defined by the expected cumulative discounted reward value. Thus, under the optimal policy $\pi^*(z_t)$, a recursive optimal $V^{\pi^*}(z_t)$ based on any initial state z_0 can be expressed as:

$$\begin{aligned} V^{\pi^*}(z_t) &= \mathbb{E}_\pi\left[R^t_{\max} \mid z_0 = z_t\right] \\ &= \max_\pi \mathbb{E}_\pi\left[\left(\sum_{t=0}^{\infty}\gamma^t r_t\right) \mid z_0 = z_t\right] \\ &= \max_\pi \mathbb{E}_\pi\left[\left(r(z_t, d_t) + \sum_{t=1}^{\infty}\gamma^t r_t\right) \mid z_0 = z_{t+1}\right] \\ &= \max_\pi\left[r(z_t, d_t) + \gamma \sum_{z_{t+1}} p_{z_t z_{t+1}}(d_t) V^{\pi^*}(z_{t+1})\right] \end{aligned} \tag{4}$$

Then, we apply the state-action function $Q(z_t, d_t)$ to estimate the $V^{\pi^*}(z_t)$ under the state z_t, and the relationship between them is $V^{\pi^*}(z_t) = \max_\pi Q^\pi(z_t, d_t)$. Therefore, the expected cumulative reward after conducting an action d_t is:

$$Q^\pi(z_t, d_t) = r(z_t, d_t) + \gamma \sum_{z_t+1} p_{z_t z_{t+1}}(d_t) V^{\pi^*}(z_{t+1}) \tag{5}$$

It iteratively approximates a Q-function $Q(z_t, d_t)$ of each state-action pair instead of modeling the dynamic knowledge of MDP. The agent will select and execute an admissible action d_t with the highest Q value under the state z_t. The core of Q-learning is the process of value iteration, and the iterative formula can be obtained as follows:

$$\begin{aligned} Q(z_t, d_t)^{new} = Q(z_t, d_t) + \beta\Big[&r(z_t, d_t) + \gamma \max_d Q(z_{t+1}, d) \\ &- Q(z_t, d_t)\Big] \end{aligned} \tag{6}$$

where $\beta \in (0,1)$ is the learning rate parameter.

B. Markov Game Model

As described above, the process of cache replacement in an available RSU can be formulated as an MDP. Nevertheless, conventional RL has not considered the influence of environment by other agents when a certain agent interacts separately, so that only the local sub-optimal solution can be obtained in the distributed edge caching system. Concerning this issue, we extend the MDP to a multi-agent system, and further formulate the process of cache replacement as a Markov (a.k.a. Stochastic) Game (MG) model. In MG, the optimal decision of a single agent cannot guarantee the global optimal solution of system since agents interact with each other. The agent in the multi-agent system has to observe not only its own reward, but those of others as well. Accordingly, a joint decision among

all agents is needed to maximize the total reward of the whole system.

In an MG, agents choose actions simultaneously and the agents' rewards are arbitrarily related in general. A standard formal definition of MG can be characterized by a tuple $\{N, \mathcal{Z}, D_1, \cdots, D_N, p, r_1, \cdots, r_N, \gamma\}$, where N is the number of agents; $\mathcal{Z}$ is the system state space, D_i is a discrete action space of i-th $(i = 1, \cdots, N)$ agent, the joint action space of all agents can be represented as $\mathcal{D} = D_1 \times \cdots \times D_N$; $p : \mathcal{Z} \times \mathcal{D} \times \mathcal{Z} \to [0, 1]$ is the state transition probability map, the joint action $\boldsymbol{d_t} := \left(d_t^1, d_t^2, \ldots, d_t^N\right) \in \mathcal{D}$ causes the transition from state z_t to z_{t+1} based on a probability $p_{z_t z_{t+1}}(d_t)$; $r_i : \mathcal{Z} \times \mathcal{D} \to \mathbb{R}$ is the reward function for agent i. Similar to MDP, given common state z_t, each agent i can execute an admissible joint action $\boldsymbol{d_t} \in D$ and receives the immediate reward r_t^i in new state z_{t+1}. MG also obeys the Markov property, for which the reward and next state of an agent only depends on the current state and the joint behavior of all agents [26].

C. Distributed MARL based Edge Caching Method

This part expands the above Q-learning into the MG model, and proposes a distributed MARL based edge caching method to obtain the optimal caching strategy.

$(\pi_1, \pi_2, \cdots, \pi_N)$ is used to denote the joint strategy of N agents. Similar to Q-learning, the expected cumulative discounted reward for agent i of the joint strategy under the current state z_t can be expressed by the state value function $V_i\left(z_t, \pi_1, \pi_2, \cdots, \pi_N\right) = \mathbb{E}_\pi\left(\sum_{t=0}^{\infty} \gamma^t r_t^i\right)$, and the objective for each agent is to learn a local π_i to maximize it.

As the maximal state value $V_i\left(z_t, \pi_1^*, \cdots, \pi_N^*\right)$ of an agent cannot be done over private strategy and the return depends on the joint strategy of all agents in the MG model, the concept of Nash equilibrium becomes very important. Nash equilibrium strategy $(\pi_1^*, \cdots, \pi_i^*, \cdots, \pi_N^*)$ is considered as an optimal joint strategy of the system, of which each agent's strategy π_i^* is the best response to the others'. In other words, any agent cannot achieve a higer reward by changing to any other strategy. Such that for $\forall z_t \in \mathcal{Z}, i = \{1, \cdots, N\}$, we have:

$$V_i\left(z_t, \pi_1^*, \cdots, \pi_i^*, \cdots, \pi_N^*\right) \geq V_i\left(z_t, \pi_1^*, \cdots, \pi_i, \cdots, \pi_N^*\right) \tag{7}$$

where $\forall \pi_i \in \Pi_i$, Π_i is the set of strategies for agent i.

Then we redefine Q-function $Q_i\left(z_t, d_t^1, \quad \cdots, d_t^N\right)$ as the expected sum of discounted reward for agent i under the state z_t. Specially, we refer to $Q_i^*\left(z_t, d_t^1, \quad \cdots, d_t^N\right)$ as Nash Q-function for agent i when all agents follow specified Nash equilibrium strategies. That is,

$$\begin{aligned} & V_i\left(z_t, \pi_1^*, \cdots, \pi_N^*\right) \\ & = \mathrm{Nash} \cdot Q_i^*\left(z_t, d_t^1, \quad \cdots, d_t^N\right) \\ & = Q_i^*\left(z_t, d_t^1, \quad \cdots, d_t^N\right) \pi_1^*\left(z_t, d_t^i\right) \cdots \pi_N^*\left(z_t, d_t^N\right) \end{aligned} \tag{8}$$

Algorithm 1 Distributed Multi-Agent Reinforcement Learning based Edge Caching Algorithm

Input: agent set $\mathcal{N}$, state space $\mathcal{Z}$, joint action space $\mathcal{D}$, learning rate β, discount factor γ, exploration factor ϵ
Output: each agent's strategy π_i^*

Initialize: set t=0, obtain the initial stste z_0.
let the learning agent be indexed by i.
$\forall\ z \in \mathcal{Z}, \forall\ i \in \mathcal{N}, \forall\ d_i \in D_1$,
$Q_t^1\left(z, d_t^1, \cdots, d_t^i, \cdots, d_t^N\right) \leftarrow 0$.
for each episode **do**
  Choose an action d_t^i using the ϵ-greedy policy in the current z_t
  for $Q_i\left(z_t, d_t^1, \cdots, d_t^N\right) \neq Q_i^*\left(z_t, d_t^1, \cdots, d_t^N\right)$ **do**
    Observe the rewards $r_t^1, \cdots, r_t^N$; the actions $d_t^1, \cdots, d_t^N$ and the next state z_{t+1}
    Update new $Q_{t+1}^i\left(s_t, d_t^i, \cdots, d_t^N\right) := (1 - \beta_t) \cdot Q_t^i\left(z_t, d_t^1, \cdots, d_t^N\right) + \beta_t\left[r_t^i + \gamma\, \mathrm{Nash} \cdot Q_t^i\left(z_{t+1}\right)\right]$
    Let $t \leftarrow t + 1$;
  end for
  Execute cache strategy according to π_i^*
end for

and

$$\begin{aligned} & Q_i^*\left(z_t, d_t^1, \quad \cdots, d_t^N\right) \\ & = [r_t^i + \beta \sum_{z_{t+1} \in Z} p_{z_t z_{t+1}}(d_1, \ldots, d_N) V_i\left(z_{t+1}, \pi_1^*, \cdots, \pi_N^*\right) \end{aligned} \tag{9}$$

Therefore, we can rewrite Nash equilibrium with the following form:

$$Q_i^*\left(z_t\right) \pi_1^*, \cdots, \pi_i^*, \cdots, \pi_N^* \geq Q_i^*\left(z_t\right) \pi_1^*, \cdots, \pi_i, \cdots, \pi_N^* \tag{10}$$

At time slot t, each agent i executes its action under the current state. After that, it observes its own immediate reward, all other agents' actions and rewards, as well as the new state z_{t+1}. Each agent i then updates its own Q-values and calculates a Nash equilibrium $\pi_1^*\left(z_{t+1}\right), \cdots, \pi_N^*\left(z_{t+1}\right)$ for the stage game $\left(Q_t^1\left(z_{t+1}\right), \cdots, Q_t^i\left(z_{t+1}\right), \cdots, Q_t^N\left(z_{t+1}\right)\right)$. The strategies that constitute a Nash equilibrium can be the optimal behavior under the current state. Existing studies [26] [27] show that there always exists at least one Nash equilibrium point in stationary strategies. In each time slot, the iterative formula of Q-value of agent i can be obtained by Eq. (11) and each Q function will converge to the Nash Q-value through the repeated interaction. More details of the proposed method is summarized in Algorithm 1.

$$\begin{aligned} & Q_{t+1}^i\left(s_t, d_t^i, \cdots, d_t^N\right) \\ & = (1 - \beta_t) \cdot Q_t^i\left(z_t, d_t^1, \cdots, d_t^N\right) + \beta_t\left[r_t^i + \gamma\, \mathrm{Nash} \cdot Q_t^i\left(z_{t+1}\right)\right] \end{aligned} \tag{11}$$

V. Performance Evaluation

In this section, we illustrate the performance of the proposed MARL-empowered vehicular edge caching and content delivery method through extensive simulations. Specifically,

TABLE I
SIMULATION PARAMETERS

Parameter	Definition	Value
N	The number of RSUs in the system	5
R	The initial cache capacity of RSU	100MB
P_i	The transmission power of RSUs	38dBm
σ^2	The power of backgound noise	-95 dBm
B_n	The bandwidth of wireless links	10MHz
W	The bandwidth of optical fibre link	20MHz
s_f	The size of requested content	[4, 12] Mbit
T_2	The delay through RSU-RSU link	10ms
T_3	The delay through RSU-MBS link	60ms

we anayze the superiority of the proposed method over other benchmark methods.

A. Simulation Setup

We consider a vehicular network consists of one MBS and 5 RSUs where the coverage area of each RSU is 200 m. The initial cache capability of RSU is set to 100 MB, and the transmission power for content delivery of RSUs is 38 dBm. It should be noted that the model can be easily generalized to scenarios where the cache capability of RSU is inconsistent. There are 10, 000 contents in the system and the size of each content is within the range of [4, 12] Mbit. The content popularity is modeled by a MZipf distribution with $\theta = 0.75$. Similar to [19], the channel gain is modeled as $30.6 + 36.7\log_{10}(d)$, and the noise power σ^2 = -95 dBm. The bandwidths of wireless links and optical fibre link are 10 MHz and 20 MHz, respectively. The transmission delay between the MBS and the RSU is 60 ms, while the transmission delay between cooperative RSUs is 10 ms. Accordingly, the price of unit leased communication resource of wireless links and optical fibre link are 0.005$ / MB and 0.009$ / MB, respectively. Detailed evaluation parameters are summarized in Table I.

For performance comparison, we introduce the following three benchmark algorithms:

1) Independent Reinforcement Learning (IRL) : IRL ignores the the multi-agent nature in MDP. Each agent learns and makes decision separately through the repeated interaction with the unknow environment, without considering the influence of other agents.
2) Least Frequently Used (LFU) [28]: When the cache capacity of each RSU is full, replace the content with the least requested times firstly.
3) Least Recently Used (LRU) [29]: When the cache capacity of each RSU is full, replace the least recently used content firstly.

Besides, for quantitatively evaluating the performance of our proposed method, the following performance metrics are used. 1) The total access cost: the average total access cost of all requested contents for each time slot. 2) Edge hit rate: the rate of requested contents that are served by RSUs instead of the remote MBS. 3) Average delay: the average access delay for all requested contents within a time slot.

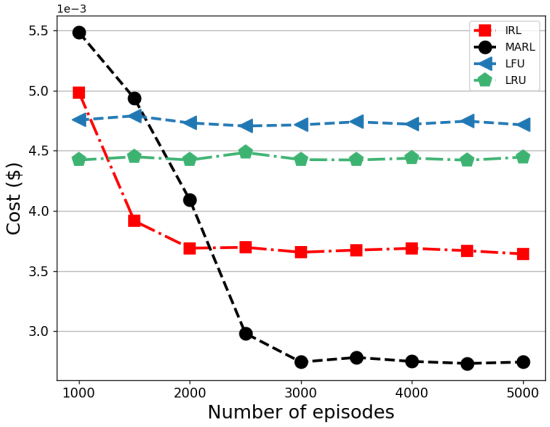
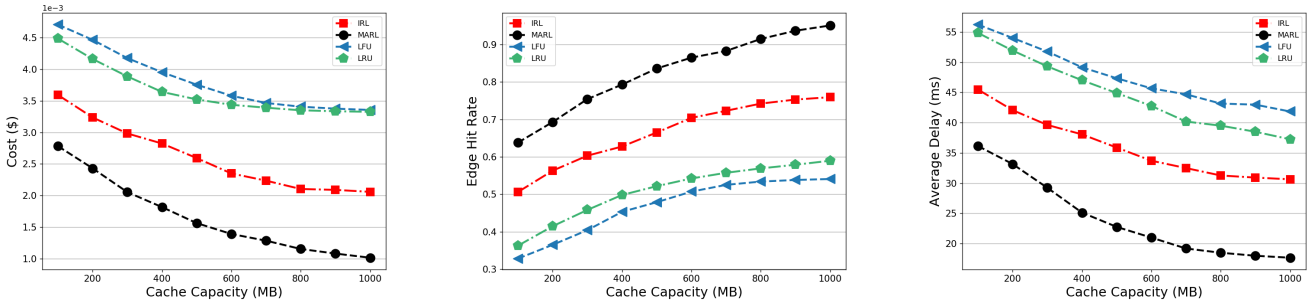


Fig. 3. Content access cost versus the number of epsiodes.

B. Performance Comparison

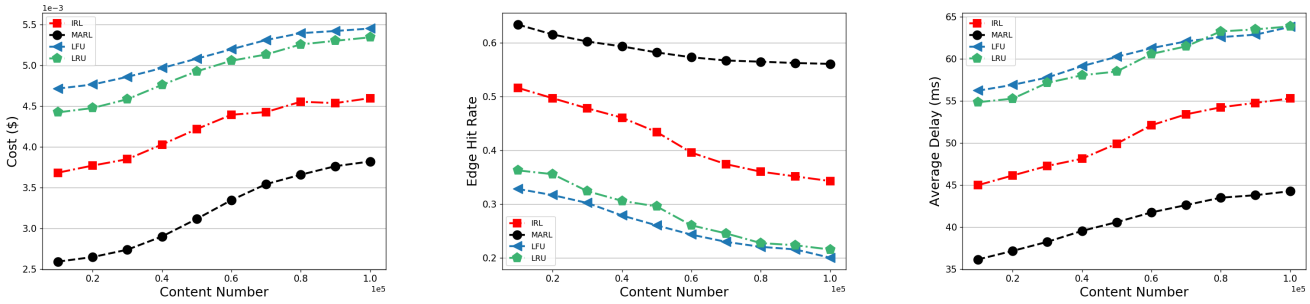
1) The convergence performance: Figure 3 shows the total content access cost versus the number of episodes when the cache capability of each RSU is fixed as 100 MB and the number of contents is 10, 000 in the system. It can be observed that for the IRL based edge caching method and the proposed MARL based edge caching method, the total content access cost of each episode decreases rapidly and gradually maintains in a relatively stable value with the increase of training episodes. Meanwhile, for the rule-based LFU and LRU method, there are no significant changes occurred over total content access cost with the increase of episodes. Overall, the proposed MARL based edge caching method performs best as it converges to a relatively stable value after running about 2500 episodes. Although the convergence speed of MARL is slightly slower than that of IRL, it can reduces about 51% total access cost in the training episodes and then obtains a lower value compared to the centralized IRL.

2) The impact of cache capability of RSUs: Then, we compare the content access cost, the average delay and the edge hit rate for different cache capability of RSUs. We change the cache capacity of each RSU from 100MB to 1000MB and display the results in Fig. 4 (in this case, the number of contents in the system is 10, 000). As shown in Fig. 4 (a) and (c), it can be observed that the proposed MARL based edge caching method always shows the best performance over other baseline methods in terms of content access cost and average delay with the increase of the cache capacity of RSUs. Specifically, it achieves the lowest average delay of 36.2ms in the initial stage, reducing about 32% total access cost compared with the state-of-the-art IRL, not mentioning the simple rule-based method LFU and LRU. Besides, the total content access cost and the average delay both have a decreasing trend for all caching methods as the cache capacity increases. This is reasonable because larger cache capacity enables RSUs to cache more popular contents simultaneously under this situation, so that most content requests can be



(a) (b) (c)

Fig. 4. (a) Content access cost versus the cache capacity of RSUs. (b) Edge hit rate versus the cache capacity of RSUs. (c) Average Delay versus the cache capacity of RSUs.



(a) (b) (c)

Fig. 5. (a) Content access cost versus the number of contents. (b) Edge hit rate versus the number of contents. (c) Average Delay versus the number of contents.

satisfied by edge RSUs.

Similarly, the increasing cache capacity has positive impacts on the metric of the edge hit rate. As shown in Fig. 4(b), it can be found that the edge hit rate increases with the increase of the cache capacity of each RSU for all caching methods, especially for our proposed MARL based edge caching method. This is because MARL jointly considers both local state and the influence of environment by other agents' behavior. In general, the proposed MARL based edge caching method will sacrifice some local hit rate and share a portion of cache capacity to process the content requests from other RSUs. This tradeoff actually reduces the average delay and the total access cost since MBS processing can be effectively avoided. In particular, the proposed MARL based edge caching method outperforms IRL, LFU and LRU caching methods with the improvement on the edge hit rate approximately 19%, 44% and 36%, respectively when the cache capacity is up to 1, 000 MB. It is worth noting that the performance of the proposed MARL based edge caching method is not always better when the cache capacity of RSUs is larger in real edge caching system. In order to reduce the resource usage cost, the available cache cooperation among RSUs should be fully utilized and considering too large cache capacity is not so cost-efficient.

3) The impact of the number of contents in the system: In the following, we change the number of contents to compare the content access cost, the average delay and the edge hit rate of different methods. We vary the number of contents from 10, 000 to 100, 000 and the initial cache capacity of RSUs is 100 MB. As shown in Fig. 5, the total content access costs and the average delay of the four methods all marginally increase with the increase of the number of contents. This is because more popular contents need to be cached in RSUs with the increase of the number of contents, which naturally causes frequent cache replacement as the cache capacity is finite. As expected, the proposed MARL based edge caching method performs well with various numbers of contents. As shown in Fig. 5 (a) and (c), it can be found that even when there are massive contents in the system (100, 000), the proposed MARL based edge caching method can still reduce 25%, 46%, 45% average delay and 21%, 39%, 42% access cost compared with IRL, LRU, and LFU caching methods, respectively. To this end, it can be inferred that the proposed MARL based edge caching method turns out to be effective under different number of contents.

Besides, for the edge hit rate, the trends of the curves are exactly the opposite. When the number of contents increases, the edge hit rate decreases for all methods. Among them, LFU and LRU still perform worst, and the corresponding edge hit

rates are only 20% and 22% respectively when the number of contents increases to 100, 000. This may due to the fact that LFU and LRU caching methods learn only from one-step past and operate based on simple rules, while IRL and MARL based edge caching methods can learn from the history of observed content demands and concentrate more on the reward that agents can obtain rather than users' requests. In addition, the variation of edge hit rate also demonstrates the aforementioned analysis, that the proposed MARL based edge caching method can better harness the utilization of available cache resources. Hence, the edge hit rate degrades much more slowly with the increase of the number of contents. With the increase of the number of contents in the system, only about 7% reduction of edge hit rate is observed in the proposed MARL based edge caching method, which also shows the stability of the proposed method.

To summarize, it can be found that the proposed MARL based edge caching method can not only reduce the total cost of content delivery, but also achieve desirable average delay and edge hit rate. Therefore, we demonstrate that the proposed MARL based edge caching method is effective under different scenarios.

VI. Conclusion and Future Work

In this paper, we have presented a cooperative edge caching architecture for IoVs, where the cooperation caching among multi-RSUs and MBS is utilized to reduce the content delivery cost and traffic burden in the system. We formulated the corresponding optimization problem to minimize the long-term overhead of content delivery, and extended the MDP to the multi-agent system. Then, we proposed an MARL based edge caching method to tackle the optimization problem. Simulation results demonstrate that our proposed method greatly outperforms other caching strategies in different scenarios.

In the future, we plan to consider vehicle-to-vehicle caching in the system, where each cache capability-enabled vehicle is able to act as a collaborative caching node for sharing content cache with RSUs and other vehicles.

Acknowledgment

This work was supported in part by NSFC grant (61872221), and NSF grants (CNS 1824440, CNS 1828363, CNS 1757533, CNS 1629746, CNS1651947, and CNS 1564128). The corresponding author is H. Zhou.

References

[1] N. Abbas, Y. Zhang, and A. Taherkordi, "Mobile edge computing: A survey," *IEEE Internet of Things J.*, vol. 5, no. 1, pp. 450-465, Feb. 2018.

[2] Z. Zhou, et al., "Social big data based content dissemination in internet of vehicles," *IEEE Trans. Ind. Informat.*, vol. 14, no. 2, pp. 768-777, Feb. 2018.

[3] S. Wang, et al., "A survey on mobile edge networks: convergence of conputing caching and communications," *IEEE Access*, vol. 5, pp. 6757 - 6779, Mar. 2017.

[4] H. Zhou, et al., "DRAIM: A Novel Delay-constraint and Reverse Auction-based Incentive Mechanism for WiFi Offloading," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 4, pp. 711-722, 2020.

[5] X. Wang, et al., "Cache in the air: Exploiting content caching and delivery techniques for 5g systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 131-139, 2014.

[6] Z. Sanaei, et al., "Heterogeneity in mobile cloud computing: taxonomy and open challenges," *IEEE Commun. Surv. Tuts.*, vol. 16, no. 1, pp. 369-392, 1st Quart. 2014.

[7] J. P. Sheu, and Y. C. Chuo, "Wildcard Rules Caching and Cache Replacement Algorithms in Software-Defined Networking," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 1, pp. 19–29, Mar. 2016.

[8] W. Zhang, et al., "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569-4581, Sep. 2014.

[9] H. Zhou, et al., "Predicting Temporal Social Contact Patterns for Data Forwarding in Opportunistic Mobile Networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10372-10383, 2017.

[10] W. Jiang, et al., "Multi-Agent Reinforcement Learning for Efficient Content Caching in Mobile D2D Networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 3, pp. 1610-1622, Mar. 2019.

[11] J. Hachem, et al., "Content caching and delivery over heterogeneous wireless networks," *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pp. 756–764, 2015.

[12] B. Blaszczyszyn, and A. Giovanidis, "Optimal geographic caching in cellular networks," *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 3358–3363, Jun. 2015.

[13] M. Yan, et al., "Assessing the energy consumption of proactive mobile edge caching in wireless networks," *IEEE Access*, vol. 7, pp. 104394 - 104404, Feb. 2019.

[14] Z. Zhao, et al., "Cluster content caching: An energy-efficient approach to improve quality of service in cloud radio access networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1207-1221, May 2016.

[15] C. Ma, et al., "Socially Aware Caching Strategy in Device-to-Device Communication Networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4615-4629, May 2018.

[16] C. Yang, et al., "Efficient Mobility-Aware Task Offloading for Vehicular Edge Computing Networks," *IEEE Access*, vol. 7, pp. 26652 - 26664, Feb. 2019.

[17] A. Sadeghi, et al., "Reinforcement learning for adaptive caching with dynamic storage pricing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2267-2281, Aug. 2019.

[18] Y. Dai, et al., "Artificial intelligence empowered edge computing and caching for internet of vehicles," *IEEE Wireless Commun. Mag.*, vol. 26, no. 3, pp. 12-18, Jun. 2019.

[19] D. Li, et al., "Deep Reinforcement Learning for Cooperative Edge Caching in Future Mobile Networks," *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, pp. 1-6, Apr. 2019.

[20] G. Qiao, et al., "Deep Reinforcement Learning for Cooperative Content Caching in Vehicular Edge Computing and Networks," *IEEE Internet of Things J.*, vol. 7, no. 1, pp. 247-257, Jan. 2020.

[21] C. Zhong, et al., "Deep Multi-Agent Reinforcement Learning Based Cooperative Edge Caching in Wireless Networks," *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 1-6, May 2019.

[22] Y. Dai, et al., "Deep reinforcement learning and permissioned blockchain for content caching in vehicular edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4312–4324, Apr. 2020.

[23] N. Luong, et al., "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, May 2019.

[24] X. Li, et al., "Hierarchical Edge Caching in Device-to-Device Aided Mobile Networks: Modeling, Optimization, and Design," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1768-1785, Aug. 2018.

[25] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-learning algorithms: A comprehensive classification and applications," *IEEE Access*, vol. 7, pp. 133653-133667, Feb. 2019.

[26] J. Hu, and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *J. Mach. Learn. Res.*, vol. 4, pp. 1039–1069, 2003.

[27] J. Hu, and M. P. Wellman, "Multiagent reinforcement learning: theoretical framework and an algorithm," *Proc. Int. Conf. Mach. Learn. (ICML)*, pp. 242–250, 1998.

[28] M. Arlitt, et al., "Evaluating content management techniques for web proxy caches," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 27, no. 4, pp. 3–11, 2000.

[29] L. Breslau, et al., "Web caching and Zipf-like distributions: Evidence and implications," *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pp. 126–134, 1999.