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

Huan Zhou, *Member, IEEE*, Kai Jiang, *Student Member, IEEE*, Shibo He, *Senior Member, IEEE*, Geyong Min, *Senior Member, IEEE*, and Jie Wu, *Fellow*, *IEEE*

***Abstract*—Edge caching is a promising approach to reduce duplicate content transmission in Internet-of-Vehicles (IoVs). Several Reinforcement Learning (RL) based edge caching methods have been proposed to improve the resource utilization and reduce the backhaul traffic load. However, they only obtain the local sub-optimal solution, as they neglect the influence from environments by other agents. This paper investigates the edge caching strategies with consideration of the content delivery and cache replacement by exploiting the distributed Multi-Agent Reinforcement Learning (MARL). A hierarchical edge caching architecture for IoVs is proposed and the corresponding problem is formulated with the goal to minimize the long-term content access cost in the system. Then, we extend the Markov Decision Process (MDP) in the single agent RL to the context of a multi-agent system, and tackle the corresponding combinatorial multi-armed bandit problem based on the framework of a stochastic game. Specifically, we firstly propose a Distributed MARL-based Edge caching method (DMRE), where each agent can adaptively learn its best behaviour in conjunction with other agents for intelligent caching. Meanwhile, we attempt to reduce the computation complexity of DMRE by parameter approximation, which legitimately simplifies the training targets. However, DMRE is enabled to represent and update the parameter by creating a lookup table, essentially a tabular-based method, which generally performs inefficiently in large-scale scenarios. To circumvent the issue and make more expressive parametric models, we incorporate the technical advantage of the Deep-Q Network into DMRE, and further develop a computationally efficient method (DeepDMRE) with neural network-based Nash equilibria approximation. Extensive simulations are conducted to verify the effectiveness of the proposed methods. Especially, DeepDMRE outperforms DMRE, Q-learning, LFU, and LRU, and the edge hit rate is improved by roughly 5%, 19%, 40%, and 35%, respectively, when the cache capacity reaches 1, 000 MB.**

***Index Terms*—Edge caching; internet-of-vehicles; content delivery; cache replacement; multi-agent reinforcement learning.**

This work was supported in part by the National Natural Science Foundation of China under Grant 62172255 and Grant U1909207; in part by the Outstanding Youth Program of Hubei Natural Science Foundation under Grant 2022CFA080. *(Corresponding author: Kai Jiang.)*

H. Zhou is with the School of Computer Science, Northwestern Polytechnical University, Xi'an 710129, China, and also with the College of Computer and Information Technology, China Three Gorges University, Yichang, 443002, China. (e-mail: zhouhuan117@gmail.com).

K. Jiang is with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430000, China. (e-mail: kai.jiang@whu.edu.cn).

S. He is with the College of Control Science and Technology, Zhejiang University, Hangzhou 310027, China (e-mail: s18he@zju.edu.cn).

G. Min is with the Department of Computer Science, College of Engineering, Mathematics, and Physical Sciences, University of Exeter, Exeter, EX4 4QF, U.K. (e-mail: g.min@exeter.ac.uk).

J. Wu is with the Department of Computer and Information Sciences, Temple University, Philadelphia PA19122, USA. (e-mail: jiewu@temple.edu).

I. Introduction

With the explosive growth of vehicles on the road and the progress of wireless multi-access technology, we have witnessed unprecedented changes in the traditional transportation system, which has evolved from a technology-driven era to a more powerful data-driven intelligent era [1]. As a fundamental paradigm of 5G networks, Internet-of-Vehicles (IoVs) enables a wide variety of vehicles to provide reliable vehicular multimedia services, cooperative cruise control, and path navigation, etc., which have paved a path towards intelligent transportation, and improved the driving safety and travel comfort of passengers [2], [3].

Meanwhile, these flourishing vehicular applications require vehicles to access vast amounts of Internet data, especially for some delay-sensitive contents, leading to severe network congestion and considerable delay in content delivery [4]–[6]. Technically, tight Quality-of-Service (QoS) requirements are generally placed for such applications, which makes cloud-based processing architectures infeasible due to the long transmission distance and limited backhaul link capacity inherent in content delivery from remote cloud servers. Indeed, the rapid increase in delay and unreliability driven by the transmission distance has become an inevitable issue for supporting massive content delivery and gained widespread attention lately [7], [8]. Despite the continuous efforts made on improving the backhaul link capacity through advanced technologies, the radio spectrum utilization has obviously reached the theoretical bound [9]–[11]. Therefore, these efforts are insufficient to cope with these grant challenges. A fundamental innovation that breaks through the bottleneck of massive content delivery in IoVs there is urgently required.

Fortunately, large-scale data analysis shows that different contents often require different priorities. Only a few popular contents account for the majority of downloads, while the access demand for most of the rest is relatively small [12], [13]. This request pattern promotes the implementation of edge caching technology in IoVs. Expressly, edge caching has provided an alternative to alleviate the backhaul link strain and content access delay inside the future IoVs, which pre-caches frequently-used contents close to vehicles by pushing cloud functions to intermediate Roadside Units (RSUs). Hence, edge caching enables vehicles to access popular content from the caching-enabled nearby RSUs directly, instead of repeatedly downloading from remote cloud servers [14]. In this way, the redundant traffic and transmission resource consumption at both backhaul and core networks can be significantly reduced, and the QoS is improved as well. In addition, enabling edge caching in IoVs can benefit extra technical superiorities from its distributed architectures and small-scale nature, including privacy protection, scalability, context awareness, and so on.

Compared with the cloud server, the cache capacity of a single edge node is insufficient to support all popular contents. Without edge cooperation, the overall cache utilization and effectiveness are prone to be under-utilized. Specifically, adjacent RSUs can share data traffic and exchange popular contents rather than operating individually, so the bilateral synergy between their caches and the centralized cloud should also be leveraged to facilitate good performance, which largely depends on a well-designed cooperative edge caching strategy. On the other hand, with the revival of artificial intelligence, Reinforcement Learning (RL) has exhibited its particular potential for the method design of efficient edge caching in IoVs. In RL, the agent can well capture the hidden dynamics of the environments and imitate the best features by trial-and-error interaction, thus learning a series of intelligence behaviors to enhance the edge cache utilization [15]–[17]. However, massively diverse, highly vibrant and distributed edge caching contexts make most previous RL work unable to adapt as they neglected the environment's influence by other agents when an agent interacts and learns the settings independently. Therefore, only the local sub-optimal solution can be obtained in the system, not the optimal global one, especially in a long-term optimization problem [18].

Indeed, it is more reasonable and helpful to investigate from a distributed perspective under the multi-agent context, which leads to inherent robustness and high scalability[1]. Meanwhile, several studies [19], [20] and [21] have proved that agents considering the impact of joint actions perform better than corresponding agents learning only based on their own actions. Therefore, this paper investigates the edge caching strategy with consideration of the content delivery and cache replacement by exploiting the distributed Multi-Agent Reinforcement Learning (MARL). A hierarchical edge caching architecture for IoVs is first proposed, where the cooperative caching among multi-RSUs and Macro Base Station (MBS) is utilized to reduce the content access cost and the backhaul traffic in the system. Then, the Markov Decision Process (MDP) in the single-agent RL is extended to the context of multi-agent system and the corresponding combinatorial multi-armed bandit problem is tackled based on the framework of stochastic games. Specifically, a **D**istributed **MAR**L-based **E**dge caching method **(DMRE)** is proposed firstly, where each agent can adaptively learn its best behaviour in conjunction with other agents for intelligent caching. Meanwhile, we attempt to reduce the computation complexity of DMRE by parameter approximation, which legitimately simplifies the training targets. However, DMRE is enabled to represent and update the parameter by creating a lookup table, essentially a tabular-based method, which generally performs inefficiently in large-scale scenarios. To circumvent the issue and make more expressive parametric models, we incorporate the technical advantage of the Deep-Q Network into DMRE, and further develop a computationally efficient method **(DeepDMRE)** with neural network-based Nash equilibria approximation. The main contributions are summarized as follows:

1) An original hierarchical edge caching architecture for IoVs is presented and the corresponding problem is formulated with the goal to minimize the long-term content access cost in the system.
2) To address the overhead minimization problem, an MDP for the optimization of cache replacement process in an available RSU is defined. Then, the MDP is extended to the context of a multi-agent system, and the corresponding combinatorial multi-armed bandit problem is tackled with the proposed DMRE firstly.
3) Furthermore, to circumvent the issue by DMRE and make more expressive parametric models, we incorporate the technical advantage of the Deep-Q Network into DMRE, and further develop a computationally efficient method (DeepDMRE) with neural network-based Nash equilibria approximation.
4) Extensive simulation results demonstrate that DeepDMRE significantly outperforms other benchmark methods in different scenarios. Meanwhile, relevant theoretical proofs of convergence and feasibility of the proposed methods are presented at the end of the paper.

The remainder of this paper is organized as follows. Section II and Section III describe the related work and the system framework, respectively. Section IV formulates the problem to minimize the long-term content access cost in the system. Section V extends MDP to a multi-agent system under the stochastic game framework, and proposes two MARL-based methods. Moreover, simulation results are analyzed in Section VI. Conclusion is summarized in Section VII.

II. Related Work

Recently, extensive works have focused on content caching strategies at the edge of networks [22]–[30]. Jedari *et al.* [22] elaborated an exhaustive review on edge caching and discussed its implementation and outlook. Wu *et al.* [23] investigated the tradeoff between content delivery delay and power consumption in small-cell network caching, and proposed an iterative algorithm to approach the optimal tradeoff between these two metrics. Yan *et al.* [24] exploited the energy assessment models for mobile edge caching and proposed a predictive caching strategy to potentially improve the cache hit rates. Zhang *et al.* [25] focused on a learning-based edge caching scheme to enable cooperation among different edge nodes with limited resources, intending to reduce the content delivery latency and maximize the overall content caching value. Similarly, to maximize the utility of the caching system, a vehicular edge caching mechanism based on user preference similarity and service availability was proposed in [26]. Furthermore, Zhao *et al.* [27] dynamically orchestrated the cache resources in IoVs by leveraging the combined power

[1]Robustness: when one or more agents fail in the system, the remaining agents can take over some of their tasks.

Scalability: by sure design, most multi-agent systems allow easy insertion of new agents.

of Lyapunov optimization, matching theory, and consensus alternating direction method of multipliers. In [28], Ao *et al.* leveraged the idea of coordinated multi-point and devised an efficient cooperative caching strategy to maximize the system throughput, where the available cache size and the network topology are both taken into account. In [29], Gu *et al.* proposed a dual-connectivity sub-6 GHz and mmWave heterogeneous network architecture to conduct a collaborative edge resource design, aiming to maximise virtual reality delivery reliability. In [30], Zhang *et al.* considered the stochastic geometry based probabilistic caching to improve the offloading rate for backhaul links, and presented an improved caching probability conversion algorithm to obtain the closed-form solutions. These works prove the advantage of edge caching. However, most of them are quasi-static and myopic, and thus perform unsatisfactorily in the diverse and dynamic IoVs system.

As Reinforcement Learning (RL) can well capture the hidden dynamics of the environments, it has been exploited to enhance the intelligence of IoVs. Specifically, Wang *et al.* [31] clustered vehicles to simplify the process of content requesting via the K-means method, and proposed an RL-based cooperative caching strategy with content request prediction in IoVs. In [32], Dai *et al.* developed a novel edge caching architecture for AI-empowered vehicular networks. They formulated a joint optimization of edge computing and caching to maximize the system utility, and exploited an RL based method to dynamically allocate computing and caching resources. Without any information or assumptions about the environment, Tan *et al.* [33] achieved an RL based adaptive framework to tackle the resource allocation of communication, caching, and computing in vehicular networks. This framework was also used by Qiao *et al.* [34] to study the problem of content placement and content delivery in IoVs, which aims to reduce the system cost under the constraint of content delivery latency. Furthermore, Wang *et al.* [35] proposed a federated deep RL-based cooperative edge caching framework to eliminate duplicate traffic and improve the edge cache utilization with specific QoS requirements. Tian *et al.* [36] considered the mobility and changing requests of self-driving vehicles, and proposed a deep RL-based collaborative caching method to break the curse of high dimensionality in the state space of MDP, as well as decrease the redundant transmission delay.

The aforementioned studies are based on centralized methods that may not always be feasible in practice due to distributed and cooperative network topology. The fundamental problem is that these methods treat the other agents as a part of the environment and thus neglect the influence on the environment by other agents' behavior. Different from them, this paper investigates the edge caching strategy considering the content delivery and cache replacement in a distributed perspective. Specifically, the DMRE framwork is proposed to reduce the backhaul traffic in the system, where each agent can adaptively learn its best behavior in conjunction with other agents for intelligent caching. In addition, to further circumvent the issue by DMRE and make more expressive parametric models, another computationally efficient method (DeepDMRE) is also proposed, which approximates the Nash equilibria by constructing neural networks.

TABLE I
NOTATIONS AND SYMBOLS

Notation	Explanation
$\mathcal{N}$	The set of all RSUs
$\mathcal{F}$	The index set of available contents
s_f	The size of content f
G_i	The limited available cache capacity of RSU i
$\mathcal{U}$	The set of all requested vehicles
ρ_f	The content popularity of content f
$T_{u,i}$	The content transmission delay between vehicle u and RUS i
$T_{i,j}$	The content transmission delay between RSU i and RSU j
$T_{i,N+1}$	The content transmission delay between RSU i and the MBS
$r_{u,i}$	The wireless downlink transmission rate between vehicle u and RSU i
$C_{u,i}$	The incurred transmission cost between vehicle u and RSU i
$C_{i,j}$	The incurred transmission cost between RSU i and RSU j
$C_{i,N+1}$	The incurred transmission cost between RSU i and the MBS
$\mathcal{L}_{f,i,j}$	The total overhead through different links
$a^t_{f,i,j}$	The caching state of local RSU i for the requested content f
$c^t_{f,i}$	The content replacement decision of RSU i in time slot t
$q^t_{i,\mathcal{F}}$	The content request arrived from the vehicles in time slot t

III. Distributed Vehicular Edge Network Architecture

This section presents the cooperative edge caching-supported IoVs architecture, including network infrastructure, content popularity pattern, as well as the process of content delivery and cache replacement. The main notations with their descriptions are listed in Table I.

A. Network Architecture

In this paper, a cooperative edge caching-supported IoVs architecture with an MBS and N small cells is considered, each of which is naturally equipped with an RSU, as shown in Fig. 1. The set of these RSUs is denoted by $\mathcal{N} = \{1, 2, \ldots, N\}$. Considering the abundant cache capacity, it is assumed that MBS (denoted by $N + 1$ in the system) is connected to the Service Providers (SPs) and can cache all contents. Here, let $\mathcal{F} = \{1, 2, \ldots, F\}$ represent the content library with total F available contents that are supported by the SPs, and s_f is the size of content $f (f \in \mathcal{F})$. Besides, each RSU $i (i \in \mathcal{N})$ is endowed with a limited available cache capacity of G_i, such that a small portion of popular contents can be cached in the RSUs to eliminate the redundant traffic. All RSUs in a cluster form are connected to the remote MBS via wired lines. U vehicles (denoted as $\mathcal{U} = \{1, 2, \ldots, U\}$) are randomly scattered in the wireless coverage scope and request various contents frequently via cellular links. Overlapping coverage between neighbouring cells is not considered, and these vehicles are assumed to be located in at least one small cell and will be served by the RSU therein; the set of vehicles in a small cell may change dynamically due to the vehicle mobility. Each RSU serves the local content requests within

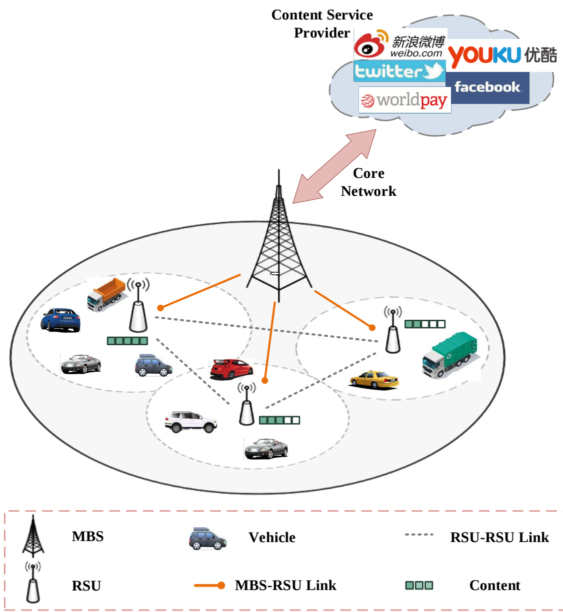


Fig. 1. Cooperative Edge Caching-supported IoVs Architecture.

its coverage range. Meanwhile, the system is assumed to operate in a fixed length of time slots (i.e., time sequence) $t \in \{1, \ldots, \Gamma\}$, where Γ denotes the finite time horizon. In each time slot, a vehicle can only request one content, while the caching decision of each RSU is also updated periodically.

As vehicles may exhibit different preferences for popular content, we introduce content popularity $\rho_f (f \in \mathcal{F})$ to reflect the content access requirements in the content library, i.e., ρ_f denotes the conditional probability of requesting content f among various vehicles' content requests. Inspired by existing works [27] [37], the content popularity is assumed to obey the Mandelbrot-Zipf distribution here. Then, the expected requesting probability for content f is obtained as:

$$\rho_f = \frac{\left(R_f + \tau\right)^{-\vartheta}}{\sum_{i \in \mathcal{F}} \left(R_i + \tau\right)^{-\vartheta}}, \quad \forall f \in \mathcal{F}, \tag{1}$$

where R_f is the rank of content f in the descending order of content popularity; $\vartheta > 0$ and τ are the skewness factor and plateau factor characterizing the distribution, respectively. Notably, due to the vehicle mobility, the local content popularity can be highly spatio-temporally varying over time slots and the requested vehicles in a cell, making caching strategy design more challenging.

Within the coverage of MBS, each RSU is able to cache various contents to meet the content access requirements from vehicles. Caching contents in RSUs enables the content request to be accommodated in proximity, without duplicate downloading from remote MBS. Particularly, each RSU can determine where the content request is answered and which local cache should be replaced. Once a vehicle sends an access request within the range of the small cell, the local RSU will traverse its cache space firstly to check whether the desired content is cached in itself. If the content is not sought in its cache, the local RSU can either retrieve the content from the adjacent RSUs or download the content directly from the MBS, and deliver it to the requested vehicle later. Meanwhile, all RSUs may replace their cache with popular contents according to the content requests of each time slot. Underlying these is a sequential decision-making problem, which corresponds to making efficient content delivery and cache replacement decisions under constraints.

B. Content Delivery Model

$T_{u,i}$, $T_{i,j}$, and $T_{i,N+1}$ are used to denote the content transmission delay between vehicle u and RSU i, RSU i and RSU j, RSU i and MBS, respectively. It is worth noticing that the transmission delay of sending request identifier by the vehicle is neglected due to its tiny data size and the high link rate in most instances. Specifically, transmission delay of content f from any requested vehicle u $(u \in \mathcal{U})$ to its connected local RSU i can be calculated as $T_{u,i} = s_f / r_{u,i}$, where $r_{u,i}$ is the wireless downlink data rate between vehicle u and RSU i. Here, we assume that each RSU consists of multiple channels, and the allocated bandwidth to each channel is the same [38]. Therefore, considering the large-scale fading, the wireless downlink data rate $r_{u,i}$ is derived by

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

where $B_{u,i}$ denotes the channel bandwidth, P_i denotes the transmission power of RSU i, σ^2 and $g_{u,i}$ are the white Gaussian noise power and the wireless propagation channel gain, respectively. Interference management is also considered for this wireless communication scenario. Furthermore, since the communications of MBS-RSU and RSU-RSU are via wired optical cables, the transmission delay $T_{i,N+1}$ and $T_{i,j}$ are set to constant values [35].

Delivering desired contents to vehicles will introduce extra transmission costs for caching nodes (cooperative RSUs or the MBS, denoted as $\mathcal{H} = \mathcal{N} \cup \{N+1\}$). As illustrated in Fig. 2, three components of transmission cost exist due to different cache hit situations. Here, $C_{u,i}$, $C_{i,j}$, and $C_{i,N+1}$ are used to denote the incurred transmission cost through RSU-Vehicle, RSU-RSU, and MBS-RSU links, respectively. Specifically, the incurred transmission cost through the RSU-Vehicle link is equal to the product of the link bandwidth, the content delivery delay, and the price of the RSU-Vehicle link's unit leased communication resource, which is expressed as:

$$C_{u,i} = B_{u,i} T_{u,i} \xi_{u,i}, \tag{3}$$

where $\xi_{u,i}$ represents the price of unit leased communication resource of wireless link. Similarly, the incurred transmission cost through RSU-RSU and MBS-RSU links can also be obtained by this way.

Here, the binary decision variable $a^t_{f,i,j} \in \{0,1\}$, $j \in \mathcal{H}$ is used to denote the caching state of any local RSU i for the requested content f in a given time slot t. When a content request occurs, the aggregate overhead of different links can be analyzed as follows:

- $a^t_{f,i,i} = 1$ indicates that content f is cached in the local RSU i at time slot t and can be delivered to the vehicle

directly. In this case, the total overhead $\mathcal{L}_{f,i,i}$ is just the transmission cost $C_{u,i}$ from the local RSU to the vehicle;
- $a^t_{f,i,j} = 1$ $(j \in \mathcal{N}\backslash\{i\})$ indicates that content f is not cached in the local RSU i, but at least one of RSUs in the system has cached the desired content, thus the local RSU can inquire and obtain it from the cooperative RSU j. Then, the total overhead $\mathcal{L}_{f,i,j}$ consists of the transmission cost $C_{i,j}$ from the cooperative RSU j to the local RSU i and the transmission cost $C_{u,i}$;
- $a^t_{f,i,N+1} = 1$ indicates that there is no expected content in all cooperative RSUs, and the local RSU i has to forward the request identifier to MBS for processing at time slot t, i.e, the local RSU i downloads content f from MBS directly. In this way, the total overhead $\mathcal{L}_{f,i,N+1}$ consists of the transmission cost $C_{i,N+1}$ from MBS to the local RSU i and the transmission cost $C_{u,i}$.

C. Cache Replacement Model

In a diverse and dynamic edge caching scenario, it is indispensable to elaborate a wise content replacement strategy for efficiently managing the cache space. In order to meet the content requests that arrive later, each cooperative RSU should update the less popular contents in each operation phase to further improve the utilization and effectiveness of the whole cache.

Generally, the local RSU needs to determine whether to keep or replace its cache contents. When the newly requested contents arrive from other adjacent RSUs or MBS, some contents in the local RSU's existing cache may be evicted because of the finite cache space. As a result, the problem is whether and which contents should be replaced with the new ones when the cache space is fully occupied. To this end, we represent the replacement control of RSU i by $\boldsymbol{c^t_{\mathcal{F},i}} = \left\{c^t_{1,i}, \ldots, c^t_{f,i}, \ldots, c^t_{F,i}\right\}$ in time slot t, where $c^t_{f,i} \in \{0,1\}$ $\quad (f \in \mathcal{F})$ indicates whether and which content in RSU i should be replaced by the current content, e.g., $c^t_{f,i} = 1$ means that content f in RSU i needs to be replaced by the current arrived one, while $c^t_{f,i} = 0$ is the opposite. To efficiently store contents in RSUs, a utility-based replacement strategy will be introduced below.

IV. Problem Formulation

This section formulates the optimization problem for IoV systems, and models the cache replacement process in an RSU as an MDP.

A. Objective Function

The RSUs in a cluster form can explore the potential of cooperation to fully utilize the cache resources. Considering the anticipated future utility, we intend to avoid myopic caching decisions while minimizing the long-term content access cost in the system. The corresponding problem is formulated as follows:

$$
\begin{aligned}
&\min_{a_{f,i,j}} \lim_{\Gamma\to\infty} \frac{1}{\Gamma} \sum_{t=1}^{\Gamma} \sum_{f=1}^{F} \sum_{i=1}^{N} \sum_{j=1}^{N+1} \rho_f a^t_{f,i,j} \mathcal{L}^t_{f,i,j} \\
s.\ t.\quad & C1: a^t_{f,i,j} \in \{0,1\}, \quad \forall f, \forall i, \forall j \\
& C2: a^t_{f,N+1,N+1} = 1, \quad \forall f \\
& C3: a^t_{f,i,j} \le a^t_{f,j,j}, \quad \forall f, \forall i, \forall j \\
& C4: \sum_{j=1}^{N+1} a^t_{f,i,j} = 1, \quad \forall f, \forall i \\
& C5: \sum_{f=1}^{F} a^t_{f,i,i} s_f \le G_i, \quad \forall i.
\end{aligned}
\tag{4}
$$

Here, $\lim\limits_{T\to\infty} \frac{1}{T}(\ldots)$ is the time averaged overhead of content delivery in the system. The meaning of the above constraints is as follows: C1 guarantees the constraint of the binary caching decision's integer nature; C2 shows the MBS's sufficient cache capacity and guarantees that MBS has cached all available contents; C3 denotes that only the cooperative RSUs or MBS, which have cached the related content, can answer the content request; C4 ensures that the content requested by a particular vehicle can only be served by an RSU or MBS ultimately; C5 is used to promise that the cache usage of each RSU should be less than its cache capacity.

B. Problem and Challenges

In fact, it is challenging to solve the above problem in Eq. (4) directly since we have to obtain the optimal coordination of the caching decision variables $a^t_{f,i,j} (f \in \mathcal{F}, j \in \mathcal{H}\backslash\{i\})$ in each time slot. However, the caching decision variable $a^t_{f,i,j}$ of any RSU is binary and time-varying, which aggravates the difficulty by solving the problem at a time slot exhaustively. The system has to collect multitudinous network state information and make the global decision on cache replacement and content delivery for each RSU. Moreover, we are devoted to a more practical case in which the prior information of the content request pattern is unknown. Thus, the optimization of the system is a combinatorial multi-armed bandit problem, and the objective function is undoubtedly NP-hard. Since the feasible set of the problem is not convex and the complexity is enormous, conventional methods using model-based heuristics or predefined rules may be unadaptable to make intelligent caching decisions under the dynamic system property.

C. Markov Decision Process

The MDP model is typically used to describe almost all sequential decision-making problems. Here, we model the optimization of cache replacement process in an RSU as an MDP. Three critical elements are identified as follows:
- **State Space:** The state in MDP is a space to reflect the IoV's environment. Accordingly, the state of an available RSU i is determined by the realization of the current caching situations and the request demand situations, which can be given as $\boldsymbol{z^i_t} = \left\{\boldsymbol{a^t_{\mathcal{F},i,i}}, \boldsymbol{q^t_{i,\mathcal{F}}}\right\}$. As described

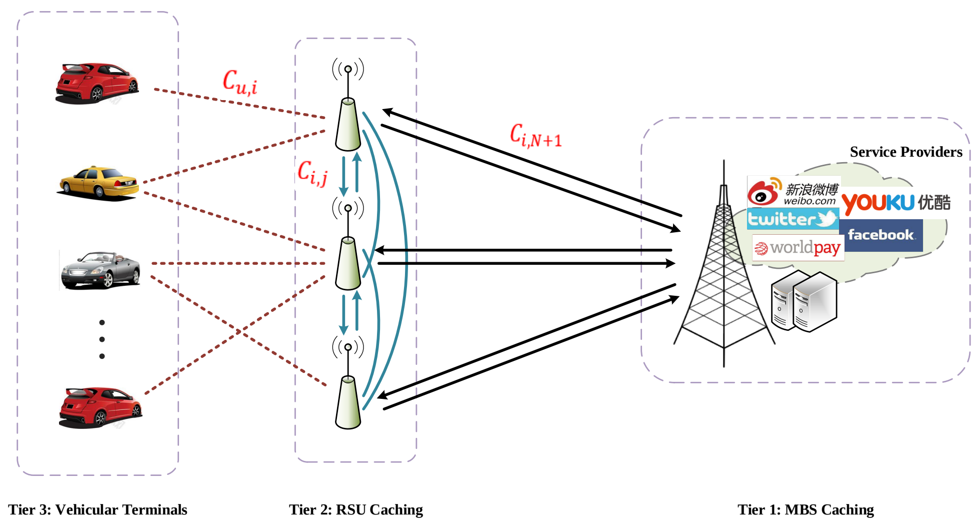


Fig. 2. Content Delivery Process and Aggregate Overhead in Hierarchical Vehicle Edge Networks.

earlier, the former $\boldsymbol{a^t_{\mathcal{F},i,i}} = \left\{a^t_{1,i,i}, \dots, a^t_{f,i,i}, \dots, a^t_{F,i,i}\right\}$ denotes the current caching state respecting to the contents in RSU i. The latter $\boldsymbol{q^t_{i,\mathcal{F}}} = \left\{q^t_{i,1}, \dots, q^t_{i,f}, \dots, q^t_{i,F}\right\}$ is the arrived content request state from vehicles in time slot t. Specifically, $q^t_{i,f} = 1$ means that at least one UE within local RSU i requests for content f in time slot t, $q^t_{i,f} = 1$ is the opposite. The above information will be aggregated as an input state in each time slot.

- **Action Space:** Once receiving a state in each time slot, the agent is responsible for deciding the caching and delivery strategy. Therefore, with the current state $\boldsymbol{z^i_t}$, the action $\boldsymbol{d^i_t}$ involves two parts, $\boldsymbol{a^t_{\mathcal{F},i,j}}$ and $\boldsymbol{c^t_{\mathcal{F},i}}$, where matrix $\boldsymbol{a^t_{\mathcal{F},i,j}}$ encodes the decision of RSU i for the requested contents, and the vector $\boldsymbol{c^t_{\mathcal{F},i}}$ indicates the content replacement control in RSU i in slot t.
- **Reward Function:** Defining an appropriate reward function is vital since the reward value is the metric for each agent to evaluate the action quality. Generally speaking, the objective function is related to the immediate reward. Our objective in the related problem is to minimize the long-term content access cost in the system, which is the inverse goal of an agent that tries to achieve the maximum cumulative discounted reward. Thus, the reward function should be negatively correlated with the optimization objective. For this purpose, the immediate reward is defined as normalized $r(\boldsymbol{z^i_t}, \boldsymbol{d^i_t}) = e^{-\sum_f^F \sum_j^{N+1} \rho_f a^t_{f,i,j} \mathcal{L}^t_{f,i,j}}$, where we utilize a negative exponential function to transform the problem legitimately. Furthermore, the reward value of an agent should satisfy the constraint conditions to ensure the validity of the results. Conversely, punition negative will be incorporated into the reward if constraints are violated in Eq. (4).

V. MARL-Empowered Cooperative Edge Caching

Compared with dynamic programming methods, it is more suitable to adopt RL-based methods when the transition probability is ambiguous in MDP. This section extends MDP to a multi-agent system under the framework of a stochastic game, and proposes two MARL-based edge caching methods.

A. Markov Game Model

According to the MDP above, we have proposed a Q-learning based edge caching method in our previous work [7]. Q-learning is a model-free independent RL method based on value iteration. Each agent in Q-learning learns the settings separately through repeated interaction, and regards other agents as a part of the environment. Although single-agent properties are transferred directly to multi-agent contexts in Q-learning based method, the IoVs system is formulated as a multitude of subsystems where the cache replacement process in each available RSU is optimized individually. Nevertheless, as an independent caching node, each RSU should have its own caching strategy according to the corresponding content request and caching state, which may be very different between different RSUs. In addition, the caching strategies of agents are mutually influential in the distributed edge caching scenarios. Q-learning based method and their variants can only obtain the local sub-optimal solution, as they do not consider the influence on the environment by other agents when an agent interacts separately, *i.e.*, each agent greedily makes decisions on its own benefit.

As far as these issues are concerned, we consider extending the MDP to a multi-agent system and formulating the cache replacement process as a Markov (a.k.a. Stochastic) Game (MG) model innovatively. In the MG model, due to the interaction between agents, the optimal decision of a single agent cannot ensure the optimal global solution of the system. Agents should not only observe their own immediate rewards and actions taken previously, but also those of others as well. Our objective is to obtain the best strategy for every agent. Therefore, it is necessary to make joint decisions among all agents to avoid the deviation in a single decision, so as to learn the coordination and optimization of the whole system.

In MG model, agents execute actions at the same time, and the rewards of agents are generally arbitrarily related.

Tuple $\{N, \mathcal{Z}, D_1, \cdots, D_N, p, r_1, \cdots, r_N, \gamma\}$ can characterize a standard formal definition of MG model, where N is the number of agents; $\mathcal{Z}$ is the finite state space of the environment, each state $z_{t+1} \in \mathcal{Z}$ in the game consists of the individual states of all agents in the system; D_i stands for a discrete action space of agent$i$ $(i = 1, \cdots, N)$; the joint action space of all agents is represented as $\mathcal{D} = D_1 \times \cdots \times D_N$, and for notational convenience, we use $\vec{d}_t \doteq \left(d_t^1, d_t^2, \ldots, d_t^N\right) \in \mathcal{D}$ to denote the current joint action of all agents in time slot t; $p : \mathcal{Z} \times \mathcal{D} \times \mathcal{Z} \to [0, 1]$ is the state transition probability function, the joint action $\vec{d}_t$ causes the transition from the game state z_t to the next new state z_{t+1} with a probability $p\left(z_{t+1}|z_t, \vec{d}_t\right)$, which should satisfy $\sum_{z_{t+1} \in \mathcal{Z}} p\left(z_{t+1}|z_t, \vec{d}_t\right) = 1, \forall z_t \in \mathcal{Z}, \forall \vec{d}_t \in \mathcal{D}$; $r_i : \mathcal{Z} \times \mathcal{D} \to \mathbb{R}$ is the direct reward function of agent i, which evaluates the quality of state transition. Given the common state z_t, each agent i can perform an acceptable joint action $\left(d_t^1, \cdots, d_t^i, \cdots, d_t^N\right)$ and accumulate a reward $r_t^i\left(z_t, \vec{d}_t\right)$ immediately in new state z_{t+1}. Notably, the reward varies not only according to the current state and the action of agent i, but also the action choice of all other agents. Besides, state transitions obey the Markov property, where the sequence of events that precede the current state cannot determine the probability distribution of future states. An agent's reward and next state depend only upon the current state and the common joint decisions among all agents [39].

B. DMRE: Distributed MARL-based Edge Caching Method

Since the distributed method is more effective in matching the edge caching system characteristics, in this part, Q-learning is expanded to the MG model, and DMRE is proposed to optimize the configuration of various caching strategies.

Indeed, the MG model can be seen as a set of matrix games associated with each state. In these games, each agent takes actions while conditioning on the behavior of others to satisfy the reward function, and $(\pi_1, \pi_2, \cdots, \pi_N)$ is used to represent the joint strategy of N agents. Similar to Q-learning, the expected cumulative discounted reward of agent i is expressed by the state value function $V_i\left(z_t, \pi_1, \pi_2, \cdots, \pi_N\right) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t^i\left(z_t, \vec{d}_t\right)]$, where $\gamma \in (0, 1)$ is the discounting factor indicating the importance of the predicted future rewards. Each agent dedicates to learning a local π_i to maximize $V_i\left(z_t, \pi_1, \pi_2, \cdots, \pi_N\right)$, meanwhile remaining robust to the actions of others over the remaining course of the game.

As the maximal state value $V_i\left(z_t, \pi_1^*, \cdots, \pi_N^*\right)$ of an agent cannot be achieved by merely private strategy, and the return explicitly depends on the joint strategy of all agents, the concept of Nash equilibrium becomes crucial. Joint Nash equilibrium strategy $\left(\pi_1^*, \cdots, \pi_i^*, \cdots, \pi_N^*\right)$ can be considered as an optimal solution of the multi-agent system, of which the game reaches the Nash equilibrium point and each agent's strategy π_i^* is the best response to the others. In other words, no agent can achieve a higher state value function by unilateral deviation, i.e., changing to any other strategy. Therefore, for $\forall z_t \in \mathcal{Z}, \forall i \in \mathcal{N}$, we formally have:

$$V_i\left(z_t, \pi_1^*, \cdots, \pi_i^*, \cdots, \pi_N^*\right) \geq V_i\left(z_t, \pi_1^*, \cdots, \pi_i, \cdots, \pi_N^*\right), \tag{5}$$

where $\forall \pi_i$ belongs to the set of strategies available to agent i.

Then, we desire a method to attain this equilibrium of the game without any prior knowledge about dynamics. Beforehand, we identify conditions that are more easily verifiable and extend the *Bellman Equation* to Nash equilibrium. Like Q-learning, we apply the dynamic programming principle to agent i's reward $V_i\left(z_t, \pi_1^*, \cdots, \pi_i^*, \cdots, \pi_N^*\right)$ while the remaining policies $\boldsymbol{\pi}^*_{\mathcal{N}\setminus\{i\}}$ are fixed, which will result in the temporal difference form as:

$$\begin{aligned} V_i\left(z_t, \pi_1^*, \cdots, \pi_N^*\right) = \max_{\pi_i} \Big[& r_t^i\left(z_t, \vec{d}_t\right) \\ & + \gamma \sum_{z_{t+1} \in \mathcal{Z}} p\left(z_{t+1}|z_t, \vec{d}_t\right) V_i\left(z_{t+1}, \pi_i \cup \boldsymbol{\pi}^*_{\mathcal{N}\setminus\{i\}}\right) \Big]. \end{aligned} \tag{6}$$

Next, different from the single-agent Q-function[2] $Q_i(z_t, d_t)$, the Q-function in the multi-agent system varies to $Q_i\left(z_t, \vec{d}_t\right)$, which is interpreted to estimate the optimal expected sum of discounted rewards (state value function) for any individual agent i during the learning process, i.e, $V_i\left(z_t, \pi_1^*, \cdots, \pi_N^*\right) = \max_{\pi_i} Q_i\left(z_t, \vec{d}_t\right)$. Specially, given the Nash equilibrium as a learning objective, $Q_i^*\left(z_t, \vec{d}_t\right)$ is referred to as a Nash Q-value for agent i when all agents follow a specified joint Nash equilibrium strategy from the next slot on. That is,

$$\begin{aligned} V_i\left(z_t, \pi_1^*, \cdots, \pi_N^*\right) &= \operatorname{Nash} Q_i\left(z_t, \vec{d}_t\right) \\ &= \pi_1^*\left(z_t, d_t^i\right) \cdots \pi_N^*\left(z_t, d_t^N\right) \cdot Q_i^*\left(z_t, \vec{d}_t\right), \end{aligned} \tag{7}$$

where $\pi_1^*\left(z_t, d_t^i\right) \cdots \pi_N^*\left(z_t, d_t^N\right) \cdot Q_i^*\left(z_t, \vec{d}_t\right)$ is a scalar, and

$$\begin{aligned} & Q_i^*\left(z_t, \vec{d}_t\right) \\ & = r_t^i\left(z_t, \vec{d}_t\right) + \gamma \sum_{z_{t+1} \in \mathcal{Z}} p\left(z_{t+1}|z_t, \vec{d}_t\right) V_i\left(z_{t+1}, \pi_1^*, \cdots, \pi_N^*\right). \end{aligned} \tag{8}$$

As a result, payoffs for each agent i are equal to Q-value at this point, and the Nash equilibrium is rewritten in the following form:

$$Q_i^*\left(z_t\right) \pi_1^*, \cdots, \pi_i^*, \cdots, \pi_N^* \geq Q_i^*\left(z_t\right) \pi_1^*, \cdots, \pi_i, \cdots, \pi_N^*. \tag{9}$$

At time slot t, each agent i executes its joint action under the current state. After that, it observes its immediate reward, all other agents' actions and rewards, as well as the new state z_{t+1}. As for agents' actions choice and Q-values update, the Nash equilibrium reward is adopt here to replace the maximum reward iteration in the single-agent method. Therefore, we rely on the stagewise approach to learn strategies separately for

[2]For ease of presentation, we have not distinguished the concept of "state-action function" and "Q-function" in this paper.

every stage game (i.e., a time slot game, defined by interim Q-values). While inequality (9) is satisfied, the stage game $\left(Q_t^1\left(z_{t+1}\right), \cdots, Q_t^i\left(z_{t+1}\right), \cdots, Q_t^N\left(z_{t+1}\right)\right)$ will be formed with the Nash equilibrium strategies $\pi_1^*\left(z_{t+1}\right), \cdots, \pi_N^*\left(z_{t+1}\right)$ under certain restrictions to Q-values' domain. To achieve this, each learning agent has to keep track and maintain a model of other agents' Q-values, and then update its own Q-values during the interaction. The strategies that constitute a Nash equilibrium can be interpreted as the optimal behavior under the current system state, while different methods for selecting among multiple Nash equilibriums will, in general, yield different updates. Existing studies [40] [41] show that there always exists at least one equilibrium point in stationary strategies, and the learning protocol provably converges if every stage game has saddle points or an optimum global; agents are mandated to update in terms of these. Ultimately, each agent i chooses its actions by calculating a Nash equilibrium, and updates the Q-value with the following iterative formula:

$$\begin{aligned} & Q_{t+1}^i\left(z_t, \vec{d}_t\right) \\ & = \left(1-\beta_t\right) Q_t^i\left(z_t, \vec{d}_t\right)+\beta_t\left[r_t^i\left(z_t, \vec{d}_t\right)+\gamma \operatorname{Nash} Q_t^i\left(z_{t+1}\right)\right], \end{aligned} \tag{10}$$

where $\operatorname{Nash} Q_t^i\left(z_{t+1}\right)$ is the Nash equilibrium of agent i under new state, $\beta_t \in(0,1)$ is the learning rate parameter, indicating how far the current estimate $Q_t^i\left(z_t, \vec{d}_t\right)$ is adjusted toward the update target $r_t^i\left(z_t, \vec{d}_t\right)+\gamma \operatorname{Nash} Q_t^i\left(z_{t+1}\right)$. Each Q-function can definitely converge to the Nash Q-value through repeated interaction when an appropriate β is designed. The agent can derive the Nash equilibrium and choose its actions accordingly.

C. Complexity Analysis and Parameter Approximation for DMRE

According to the description above, each agent i can gradually learn the equilibrium at each stage, which is determined by the joint actions of all agents in the system. That means the agent needs to learn N Q-values of all agents and derive strategies from them. Specifically, agent i will update $\left(Q^1, \cdots, Q^i, \cdots, Q^N\right)$ for all system states and joint actions, while these Q-values are updated and maintained internally by it. Hence, DMRE is often computationally expensive for equilibrium computation, and there may not be enough memory to maintain the corresponding values in large scale scenario. Consequently, it is necessary to analyze the computation complexity and further give simplification of the proposed method.

Let $|\mathcal{Z}|$ denote the number of system states, and $|D_i|$ be the size of agent i's action space D_i. Assuming $|D_1| = \cdots = |D_N| = |D|$, the total number of entries in Q^i is $|\mathcal{Z}| \cdot |D|^N$. As the agent has to maintain N Q-tables, the total space requirement is $N|\mathcal{Z}| \cdot |D|^N$. Then, the proposed method is linear in the number of states, polynomial in the number of actions, but exponential in the number of agents in terms of space complexity. On the other hand, the training time of DMRE is dominated by the calculation of the Nash equilibrium. That is, each agent will search for an optimal solution according to the current solutions of other agents while the computation process is nonlinear if the number of agents is more than two [39]. Thus, due to the remarkable efficiency and generalization, parameter approximation is introduced to reduce the computation complexity for DMRE in this paper, which legitimately simplifies the training targets.

Specifically, elements can be controlled by sequences approximately in the space that is composed of Q-functions. Correspondingly, we introduce a set of parameters θ and approximate any agent i's Q-function by using the updated parameter. The Q-function is re-expressed as:

$$\hat{Q}^i(z, \vec{d}, \theta) \approx \boldsymbol{\theta}^T \boldsymbol{\phi}_{\boldsymbol{i}}=\sum_{j=1}^N \theta_j \phi_{ij}(z), \tag{11}$$

where $\boldsymbol{\theta}=\left(\theta_1, \theta_2, \cdots, \theta_n\right)^T$ and the given eigenfunction $\boldsymbol{\phi}=\left(\phi_1(s), \phi_2(s), \cdots, \phi_n(s)\right)^T$.

Since $Q^i(z, \vec{d}) \doteq \hat{Q}^i(z, \vec{d}, \theta)$ is possible when the error is smaller, we attempt to find a set of parameters θ for which the corresponding approximate values $(\hat{Q_\theta^1}, \cdots, \hat{Q_\theta^N})$ approximately satisfy Eq. (8). Thus, the loss function is interpreted as the distance between the approximate value and true value of Q-functions, which can be converted to:

$$\begin{aligned} Loss(\theta) = & \underset{z \sim p\left(z_{t+1} \mid z_t, \vec{d}_t\right)}{\mathbb{E}}\left[\left(Q^i(z, \vec{d})-\hat{Q}^i(z, \vec{d}, \theta)\right)^2\right] \\ & =\frac{1}{T} \sum_{t=1}^T\left(r_t^i+\gamma \operatorname{Nash} \hat{Q}_t^i\left(z_{t+1}, \theta\right)-\hat{Q}_t^i\left(z_t, \vec{d}_t, \theta_t\right)\right)^2 . \end{aligned} \tag{12}$$

According to the *Bellman Equation* of the state-action functions, Stochastic Gradient Descent (SGD) is performed to train θ towards the target value by minimizing the loss function, i.e. $\operatorname{argmin}(Loss(\theta))$. Therefore, the optimal value is generated by an error as:

$$\theta_{t+1} \approx \theta_t-\frac{1}{2} \beta_t \nabla\left(r_t^i+\gamma \operatorname{Nash} \hat{Q}_t^i\left(z_{t+1}, \theta\right)-\hat{Q}_t^i\left(z_t, \vec{d}_t, \theta_t\right)\right)^2 . \tag{13}$$

We then take a semi-gradient method and ignore the derivative of parameter θ with respect to the unknown Nash Q value. Thus, the update formula of parameter θ in state-action function $\hat{Q}^i(s, \vec{d}, \theta)$ of agent i is written as follows:

$$\begin{aligned} \theta_{t+1}= & \theta_t+\beta_t\left[r_t^i+\gamma \operatorname{Nash} \hat{Q}_t^i\left(z_{t+1}\right)\right. \\ & \left.-\hat{Q}_t^i\left(z_t, \vec{d}_t, \theta_t\right)\right] \nabla \hat{Q}_t^i\left(z_t, \vec{d}_t, \theta_t\right) . \end{aligned} \tag{14}$$

Instead of updating the Q-function, parameter approximation can make it more efficient by using the updated parameter θ. Throughout the training process of seeking the Nash equilibrium point, the agent first updates the target θ by minimizing the corresponding loss function (12) via SGD. Then, θ can continually train the Q-function towards the Nash equilibrium point. All of these processes make the agent approach the Nash equilibrium point effectively. More details of the simplified DMRE[3] are summarized in **Algorithm 1**,

[3]As the simplified DMRE has lower computation complexity and is easier to implement than the original version, we only use the "DMRE" to denote the simplified version in the following.

Algorithm 1: Parameter Approximation for **DMRE**

Input: agent set $\mathcal{N}$, state space $\mathcal{Z}$, joint action space $\mathcal{D}$, learning rate β, discount factor γ, exploration factor ϵ.

Each agent $i \in \mathcal{N}$ **do**
Initialize $\forall\ z \in \mathcal{Z}, \forall\ d_i \in D_1, \hat{Q}^i(z, \vec{d}, \theta) \leftarrow 0$, parameter θ.
for each episode **do**
    Set t=0, obtain the initial state z_0 by randomly caching.
    for each slots of episode **do**
        Derive an action d_t^i based on the ϵ-greedy strategy in the current z_t.
        for $Q_i\left(z_t, \vec{d}_t\right) \neq Q_i^*\left(z_t, \vec{d}_t\right)$ **do**
            Observe the rewards $r_t^1, \cdots, r_t^N$, the joint actions $\vec{d}_t$, and the next state z_{t+1};
            Update parameter θ according to Eq. (14);
            Compute $Q^i\left(z_t, \vec{d}_t\right)$: $Q^i\left(z_t, \vec{d}_t, \theta\right) \leftarrow \theta^T \phi_i$;
            Solve Nash equilibrium strategies under state z_{t+1};
            Let $t \leftarrow t+1$.
        Execute cache strategy according to π_i^*.

and relevant theoretical proofs of convergence and feasibility are presented at the end of this paper.

D. DeepDMRE: Distributed Deep MARL-based Edge Caching Method

DMRE is enabled to represent and update the parameter by creating a look-up table, which is essentially a tabular-based method, even though parameter approximation is used to simplify the training target. Parameter update rule Eq. (14) in an agent relies on the repeated calculation of Nash $\hat{Q}_t^i(z_{t+1})$ over all states, which generally performs inefficiently in large-scale scenarios. To circumvent the issue and make more expressive parametric models, we incorporate the technical advantage of Deep-Q Network[4] into DMRE, and further develop a computationally efficient method (DeepDMRE) with neural network-based Nash equilibria approximation.

As mentioned above, we can introduce a set of parameters and approximate any agent's Q-function using Eq. (11). In contrast, DeepDMRE defines a specific model for the function approximation. For each parameter θ, we decompose $\hat{Q}^i(z, \vec{d}, \theta)$ into two components:

$$\hat{Q}(z, \vec{d}, \theta) = \hat{V}(z, \theta) + \hat{A}(z, \vec{d}, \theta), \tag{15}$$

where $\hat{V}(z, \theta)$ and $\hat{A}(z, \vec{d}, \theta)$ are produced by various neural network components. $\hat{V}(z, \theta)$ is the state value function under state z, and $\hat{A}(z, \vec{d}, \theta)$ is the advantage function of executing action under state z. The advantage function represents the optimality gap between $\hat{Q}$ and $\hat{V}$. In particular, each agent i's behaviour must be the best response to the others when the action execution constitutes a Nash equilibrium. At this point, the advantage function $\hat{A}(z, \vec{d}, \theta) = \mathrm{Nash}\, Q_i\left(z_t, \vec{d}_t\right) - V_i\left(z_t, \pi_1^*, \cdots, \pi_N^*\right) = 0$.

In contrast to DMRE, DeepDMRE maintains neural network models with experience replay buffers on each agent. The basic concept underlying DeepDMRE is centralized training and distributed inference. During training sample collection, the agent will store its observed experience tuple $y_t = (z_t, \vec{d}_t, z_{t+1}, r_t)$ into the replay buffer, which involves its current state and available action set. The arrived experience samples are used to train the parameters of the function approximation in neural networks via SGD. Practically, this replaying enables the agent to randomly extract a minibatch of previous experience samples from the replay buffer for learning at each iteration. Empirical studies [15] [20] [33] have proved that the experience replay mechanism can enhance sample efficiency and break the correlation among data training.

Moreover, we generalize and adopt the model hypothesis proposed in [40], which considers a second order Taylor expansion $\hat{Q}^i(z, \vec{d}, \theta)$ in action around the Nash equilibrium. Together with the hypothesis, the advantage function for each agent i can be approximately written in a linear-quadratic form as:

$$\begin{aligned} &\hat{A}_i(z, \vec{d}, \theta) \\ &= \left(\vec{d}_{\mathcal{N}\setminus\{i\}} - \vec{\mu}^{\theta}_{\mathcal{N}\setminus\{i\}}\right)^{\top} \mathbf{\Psi}_i^{\theta}(z) \\ &- \begin{pmatrix} d_i - \mu_i^{\theta} \\ \vec{d}_{\mathcal{N}\setminus\{i\}} - \vec{\mu}^{\theta}_{\mathcal{N}\setminus\{i\}} \end{pmatrix}^{\top} \boldsymbol{P}_i^{\theta}(z) \begin{pmatrix} d_i - \mu_i^{\theta} \\ \vec{d}_{\mathcal{N}\setminus\{i\}} - \vec{\mu}^{\theta}_{\mathcal{N}\setminus\{i\}} \end{pmatrix} \end{aligned} \tag{16}$$

where the block matrix $\mathbf{\Psi}_i^{\theta}(x)$ and $\boldsymbol{P}_i^{\theta}(x)$ are all deterministic probability distributions. Interested readers can find detailed information in [40].

This model hypothesis implicitly suggests that each agent's Q-function can be approximately expressed as a linear-quadratic function of the actions. Each $\hat{Q}^i(z, \vec{d}, \theta)$ is a concave function of d_i , guaranteeing that Nash $\hat{Q}_t^i(z_{t+1})$ is bijective. Then, we can model $\hat{V}(z, \theta)$, $\vec{\mu}^{\theta}$, and relevant probability distributions via constructed neural networks, rather than modelling $\hat{Q}^i(z, \vec{d}, \theta)$. The Nash equilibrium is achieved when $\vec{d}^* = \vec{\mu}^{\theta}$, and the advantage function will become 0 simultaneously. Similar to the derivations in Eq. (7), we can simplify expressions of the value function and equilibrium strategy as:

$$\hat{V}(z, \theta) = \mathrm{Nash}\, \hat{Q}^i(z, \vec{d}, \theta) \tag{17}$$

and

$$\vec{\mu}(z) = \mathrm{argmax}\, \mathrm{Nash}\, \hat{Q}^i(z, \vec{d}, \theta) \tag{18}$$

Consequently, the loss function in Eq. (12) becomes more tractable through this simplification. For a minibatch of J sample, it can be converted as:

[4]Deep-Q Network is a classical deep RL method based on value iteration, in which neural networks are adopted to estimate the Q-function.

Algorithm 2: DeepDMRE: Distributed Deep MARL based Edge Caching

Input: agent set $\mathcal{N}$, state space $\mathcal{Z}$, joint action space $\mathcal{D}$, learning rate β, discount factor γ, exploration factor ϵ, experience replay buffer $\mathcal{M}_i$, minibatch size J.

```
Each agent i ∈ N do
Initialize ∀ z ∈ Z, ∀ d_i ∈ D_i, Q̂^i(z, d⃗, θ) ← 0, replay
  buffer M_i, neural network parameter (θ_V, θ_A).
for each episode do
    Set t=0, obtain the initial state z_0 by randomly
      caching.
    for each slots of episode do
        Derive an action d⃗ ← μ⃗^θ based on the
          ε-greedy strategy in the current z_t.
        for Q_i(z_t, d⃗_t) ≠ Q_i^*(z_t, d⃗_t) do
            Observe the rewards r_t^1, ..., r_t^N, the actions
              d⃗_t, and the next state z_{t+1};
            Store observed experience tuple
              y_t = (z_t, d⃗_t, z_{t+1}, r_t) into the replay buffer.
            if t % updateFrequency == 0 then
                Randomly extract a minibatch of J
                  experiences {y_j}_{j=1}^J from replay
                  buffer M_i.
            Update sub-parameters θ_V and θ_A by
              minimizing L(θ) as Eq. (20);
            Perform a an actor-critic-based variant on
              the loss function with respect to the
              sub-parameters θ by ∂Loss(θ)/∂θ;
            Let t ← t + 1.
        Update (θ_V, θ_A).
```

$$\mathcal{L}(\theta) = \frac{1}{J}\sum_{j=1}^{J}\mathcal{L}_j(\theta) = \frac{1}{J}\sum_{j=1}^{J}\left(r_t^i + \gamma\hat{V}_t^i\left(z_{t+1},\theta\right) - \hat{V}(z,\theta) - \hat{A}(z,\vec{d},\theta)\right)^2. \tag{19}$$

where each sample observation $y_j = (z_j, \vec{d}_j, z_{j+1}, r_j)$ is used to train the parameter θ by miniminzing the $\mathcal{L}_j(\theta)$.

Furthermore, instead of applying SGD with back-propagation, we employ an actor-critic-based variant on the loss function, enabling more stable and efficient convergence processes. Specifically, there are N actor networks and one centralized critic network, where each RSU adopts one of the actor networks to execute its inference, and the MBS acts as a centralized critic network to train the model and evaluate the overall caching state. Since the value function can be modeled independently through the decomposition in Eq. (15), an actor-network is performed for parameters update by minimizing the loss function in Eq. (19) over parameters θ. Notably, the parameter θ gets separated as $\theta = (\theta_V, \theta_A)$ in this update rule, where sub-parameter θ_V is used to model $\hat{V}(z, \theta_V)$, and sub-parameter θ_A is used to model $\hat{A}(z, \vec{d}, \theta_A)$. Accordingly, the training objective is to update these parameters by minimizing the redefined loss function:

$$\mathcal{L}(\theta) = \frac{1}{J}\sum_{j=1}^{J}\mathcal{L}_j(\theta) = \frac{1}{J}\sum_{j=1}^{J}\hat{\mathcal{L}}(y_j, \theta_V, \theta_A), \tag{20}$$

where each agent minimizes $\mathcal{L}(\theta)$ by alternating between minimization in sub-parameters θ_V and θ_A. Here, the loss founction of an individual sample $y_j = (z_j, \vec{d}_j, z_{j+1}, r_j)$ corresponds to the error in Eq. (8), which is expressed as:

$$\hat{\mathcal{L}}(y_j, \theta_V, \theta_A) = \left(r(z_j, \vec{d}_j) + \gamma\hat{V}\left(z_{j+1}, \theta_V\right) - \hat{V}(z_j, \theta_V) - \hat{A}(z_j, \vec{d}_j, \theta_A)\right)^2. \tag{21}$$

More details of DeepDMRE are summarized in **Algorithm 2**. With the locally linear-quadratic form of the advantage function, we minimize the loss function in Eq. (21) over sub-parameters θ_V and θ_A by actor-critic-based iterative optimization and batch sampling. On the other hand, we also apply and modify the ϵ-greedy based action selection strategy for ensuring adequate exploration, where ϵ is a decreasing parameter to achieve a tradeoff between exploitation and exploration.

VI. Performance Evaluation

This section conducts extensive simulations to assess the performance of the proposed methods. Specifically, the superiority of the proposed methods is demonstrated over both rule-based and learning-based benchmarks.

A. Simulation Setup

This paper considers an IoVs architecture composed of one MBS and 5 RSUs, in which the coverage area of each RSU is 200 m. RSUs are located at the center of each region to serve the corresponding content requests, and the initial cache capability of each RSU is limited to 100 MB. Notably, the model can be easily extended to the case of inconsistent RSU cache capacity. Besides, 10, 000 contents are generated, and the size of each content is within the range of [0.5, 1.5] MB. The content popularity follows an MZipf distribution with $\theta = 0.73$. According to ref. [4] [35], we set the channel gain as $30.6 + 36.7\log_{10}(d)$, and the noise power σ^2 as -95 dBm. The transmission delays between MBS and RSU, and cooperative RSUs are 80 ms and 10 ms, respectively. Here, the price of the unit leased communication resources ($ / MB) is set as 0.005 for wireless links and 0.009 for optical links, while their bandwidths are 10 MHz and 20 MHz, respectively. Meanwhile, we set the capacity of experience replay buffer $\mathcal{M} = 500$, minibatch size J=32, and initial exploration factor $\epsilon = 0.05$. The learning rate parameters for the actor-network and the critic-network are set as 10^{-3} and 3×10^{-4}, respectively. All experiments are run on 24 core -

TABLE II
PARAMETERS SETUP

Parameter	Definition	Value
N	Default RSU number	5
T_2	Delay between any cooperative RSUs	10ms
T_3	Delay between the MBS and an RSU	80ms
P_i	Transmission power of RSUs	38dBm
B	Bandwidth of wireless links	10MHz
W	Bandwidth of optical fibre links	20MHz
σ^2	Power of background noise	-95 dBm
s_f	Size of requested contents	[0.5, 1.5] MB

2.4GHz Intel Xeon E5-2650 processor and 256GB RAM. The main parameters are listed in Table II.

For performance comparison, the following four benchmark caching methods are introduced:

1) Least Frequently Used (LFU): The content with the least requested times will be replaced firstly when an RSU's cache capacity is full.
2) Least Recently Used (LRU): The content with the longest unused time will be replaced firstly when an RSU's cache capacity is full.
3) Q-learning [7]: An independent reinforcement learning method, the process of cache replacement in each available RSU is optimized individually.
4) Informed Upper Bound (IUB) [13]: An omniscient method which assumes all of the requesting information is perfectly known in advance. It provides an upper bound of other methods and is usually hard to achieve in practice.

Besides, the following metrics are used to evaluate the performance of different methods quantitatively. 1) Content access cost: the average total access cost of all requests in each time slot. 2) Edge hit rate: the sum of requests served by RSUs divided by the total number of requests. 3) Average delay: the average access time for all requests in each time slot.

B. Performance Comparison

1) The convergence performance: We plot the convergence curve for three RL-based methods in Fig. 3, where the number of contents and the cache capability of RSU are fixed at 10, 000 and 100MB, respectively. We can observe that for these methods, each episode's total content access cost decreases rapidly with the increase of the number of episodes. The DeepDMRE converges to a relatively stable value after running about 1800 episodes. Overall, it performs best as it achieves a faster running time and obtains the same convergence value as the original DMRE. On the other hand, although the convergence speed of distributed MARL-based methods (DMRE and DeepDMRE) are slightly slower than that of the independent Q-learning based method, they can reduce the content access cost up to 50% during the training episodes.

2) The impact of cache capability of RSUs: Then, we explore the impact of different cache capabilities of RSUs

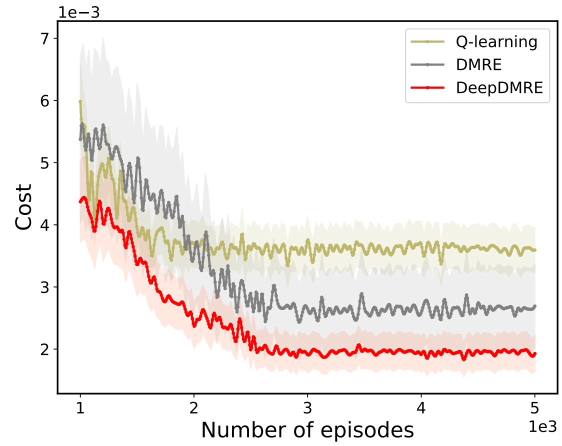
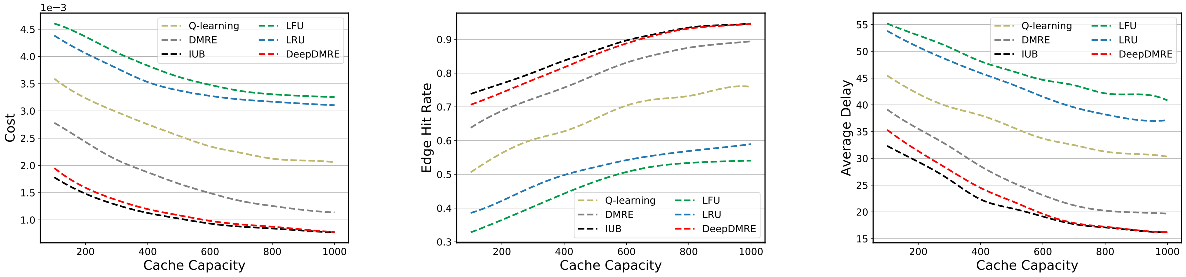


Fig. 3. Content access cost versus the number of episodes.

on the performance of different methods, as shown in Fig. 4. The cache capacity of each RSU is changed from 100 MB to 1000 MB and the number of contents is 10, 000 in this case. As expected, IUB performs best. However, it is impractical since future information cannot be perfectly known. Among the other caching methods, we observe that as the cache capacity of RSUs increases, DeepDMRE consistently achieves better performance with respect to the content access cost and average delay. Specifically, DeepDMRE achieves the lowest average delay of 33ms in the initial stage, and the content access cost is reduced by about 47% and 26% compared with independent Q-learning and DMRE, without mentioning the simple rule-based methods LFU and LRU. In addition, for all caching methods, the content access cost and the average delay decrease with the increase of the cache capacity, which is reasonable. The larger cache capacity enables the RSU to cache more contents at the same time so that RSUs can meet most content requests.

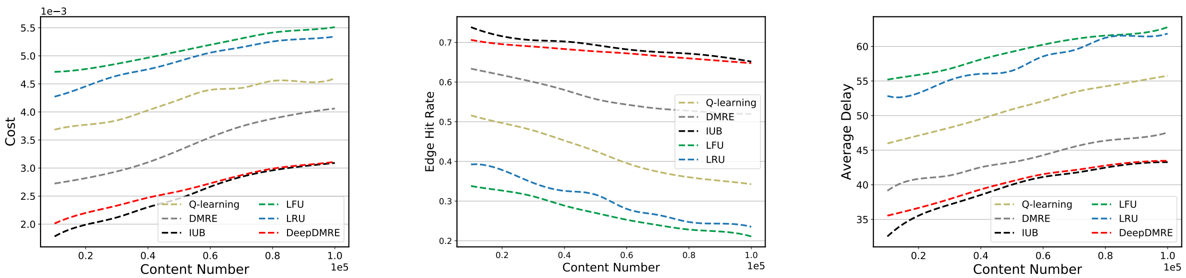
Similarly, the increase of cache capacity also has a positive impact on edge hit rate. From Fig. 4 (b), we can find that the edge hit rate exhibits an increasing trend for all caching methods when we increase the cache capacity of each RSU, especially for DMRE and DeepDMRE, which are very close to the value of IUB. This is because they considers the influence of environments by other agents, and takes joint decisions to learn the coordination and optimization of the whole system. Notably, DMRE and DeepDMRE will generally sacrifice some local hit rate and share cache capacity to serve requests from neighboring RSUs. This tradeoff benefits much by reducing the duplicate content transmission since MBS fetching is largely avoided. Especially, DeepDMRE outperforms DMRE, Q-learning, LFU, and LRU. For example, the edge hit rate is improved by roughly 5%, 19%, 40%, and 35%, respectively, when the cache capacity reaches 1, 000 MB. At this moment, content replacement processes may rarely occur in the RSUs.

3) The impact of the number of contents in the system: This part changes the number of contents to evaluate the performance of different methods. The number of contents is varied from 10, 000 to 100, 000 and the initial cache



(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.

capacity of RSUs is set as 100 MB. As illustrated in Fig. 5, the content access cost and the average delay of all methods increase slightly as the number of contents increases. The increasing trend does indicate that when the cache capacity is finite, more contents will naturally cause frequent cache replacement, as more contents need to be cached in RSUs now. As expected, DeepDMRE performs better with a different numbers of contents, except IUB. From Fig. 5 (a) and (c), we can find that even when there is a massive number of contents (i.e., 100, 000), DeepDMRE can still reduce the content access cost by 11%, 16%, 28%, 34%, and the average delay by 15%, 21%, 33%, 37% compared to DMRE, Q-learning, LRU, and LFU, respectively, which infers the effectiveness of DeepDMRE under different numbers of contents.

Besides, the edge hit ratio of all methods decreases with the increase of the number of contents. Among them, edge hit rate of DeepDMRE is 64% when the number of contents reaches 100, 000. In contrast, LFU and LRU still perform worst for this moment, and the edge hit rate is only 21% and 23%, respectively. This situation may occur because LFU and LRU learn only from one-step past and operate based on simple rules, while RL-based edge caching methods can be derived from the observed historical content demands and concentrate more on the reward that agents can earn rather than users' requests. Furthermore, the variation of edge hit rate also confirms the analysis above. That is, DeepDMRE improves the cache resources utilization significantly.

To summarize, quantitative results validate the effectiveness of DMRE and DeepDMRE under different scenarios. As observed, they not only reduce the content access cost, but also achieve a desirable edge hit rate and average delay.

VII. Conclusion

This paper first considered a cooperative edge caching supported IoVs architecture, which uses the cooperative caching between multiple RSUs and MBS to reduce the duplicate content transmission in the system. Then, the MDP was extended to the context of a multi-agent system, and the corresponding combinatorial multi-armed bandit problem was further tackled based on the framework of stochastic games. Specifically, a **D**istributed **M**A**R**L-based **E**dge caching method (DMRE) was proposed firstly, where each agent can adaptively learn its best behavior in conjunction with other agents for intelligent caching. Meanwhile, we attempted to reduce the computation complexity of DMRE by parameter approximation, which legitimately simplifies the training targets. However, DMRE is enabled to represent and update the parameter by creating a lookup table, essentially a tabular-based method, which generally performs inefficiency in large-scale scenarios. To circumvent the issue and make more expressive parametric models, we combined the technical advantage of the Deep-Q Network into DMRE, and further developed a computationally efficient method, named DeepDMRE, by constructing neural networks for approximating the Nash equilibria. Simulation results demonstrated that DeepDMRE significantly outperforms other benchmark methods in different scenarios.

Appendix

This part gives the relevant theoretical proofs of convergence and the feasibility of the simplified DMRE as follows.

A. Convergence Analysis

Since the original DMRE has a provably convergence performance with restrictive conditions [39], [41], the convergence of the simplified DMRE can accordingly be proved by demonstrating the effectiveness of SGD. Therefore, as θ (which minimizes the corresponding loss function) is gradually obtained via SGD, the state-action function (in the form of $\hat{Q}^i(z,\vec{d},\theta)$) will be updated towards the Nash equilibrium point.

For any state-action function $Q^k (k \in \mathcal{N})$ maintained internally by a learning agent, we can define $f_k(\theta) = \left(Q^k(z,\vec{d}) - \hat{Q}^k(z,\vec{d},\theta)\right)^2$. Then, the corresponding loss function is written as:

$$L(\theta) \triangleq Loss(\theta) = \frac{1}{N}\sum_{k=1}^{N} f_k(\theta). \tag{22}$$

As described in Section V-C, we train the parameter θ towards the target value by minimizing the loss function at each iteration. Then, owing to the numerical convexity of the loss function, its gradient $\nabla L(\theta)$ satisfies the $Lipschitz\ Continuity$ with respect to the constant G, so that for all θ_t and θ_{t+1}. We have:

$$\|\nabla L(\theta_t) - \nabla L(\theta_{t+1})\| \leqslant G\|\theta_t - \theta_{t+1}\|. \tag{23}$$

For a series of progressively decreasing learning rates β_t, the iterative formula of parameter θ can be expressed as:

$$\theta_{t+1} = \theta_t - \beta_t \nabla L(\theta_t). \tag{24}$$

Meanwhile, each θ in the training process should satisfy $\max_k \|\nabla f_k(\theta_t)\| \leq B\|\nabla L(\theta_t)\|$, where B is a constant. That is to say, the optimal θ for loss function is supposed to be a stagnation point at the same time. We denote the error during the iteration process as e_t, then the above Eq. (24) can be rewritten with the following form:

$$\theta_{t+1} = \theta_t - \beta_t (\nabla L(\theta_t) + e_t), \tag{25}$$

where the error $e_t = \nabla f_k(\theta_t) - \nabla L(\theta_t)$.

Additionally, since the mean of the error is equal to 0, we therefore get:

$$E[e_t] = E[\nabla f_k(\theta_t) - \nabla L(\theta_t)] = E[\nabla f_k(\theta_t)] - \nabla L(\theta_t) = 0. \tag{26}$$

Furthermore, based on the derivations above, we can easily get:

$$\begin{aligned} &E\left[\|e_t\|^2\right] \\ &= E\left[\|\nabla f_k(\theta_t) - \nabla L(\theta_t)\|^2\right] \\ &= E\left[\|\nabla f_k(\theta_t)\|^2 - 2\langle \nabla f_k(\theta_t), \nabla L(\theta_t)\rangle + \|\nabla L(\theta_t)\|^2\right] \\ &= E\left[\|\nabla f_k(\theta_t)\|^2 - 2\langle E[\nabla f_k(\theta_t)], \nabla L(\theta_t)\rangle + \|\nabla L(\theta_t)\|^2\right] \\ &= \frac{1}{N}\sum_{k=1}^{N}\left[\|\nabla f_k(\theta_t)\|^2 - \|\nabla L(\theta_t)\|^2\right] \\ &\leq \left(B^2 - 1\right)\|\nabla L(\theta_t)\|^2. \end{aligned} \tag{27}$$

Then, for inexact gradients, inequality (23) can be rewritten by incorporating Eq. (24) with it:

$$L(\theta_{t+1}) \leqslant L(\theta_t) + \langle \nabla L(\theta_t), \theta_{t+1} - \theta_t\rangle + \frac{G}{2}\|\theta_{t+1} - \theta_t\|^2. \tag{28}$$

Also, we substitute error $e_t = \nabla f_k(\theta_t) - \nabla L(\theta_t)$ and Eq. (24) into inequality (28), and further obtain:

$$\begin{aligned} L(\theta_{t+1}) \leq & L(\theta_t) - \beta_t \langle \nabla L(\theta_t), \nabla L(\theta_t) + e_t\rangle \\ & + \frac{\beta_t^2 G}{2}\|\nabla L(\theta_t) + e_t\|^2 \\ = & L(\theta_t) - \beta_t\left(1 - \frac{\beta_t G}{2}\right)\|\nabla L(\theta_t)\|^2 \\ & - \beta_t(1 - \beta_t G)\langle \nabla L(\theta_t), e_t\rangle + \frac{\beta_t^2 G}{2}\|e_t\|^2. \end{aligned} \tag{29}$$

As the learning rate β_t tends to be infinitesimal, we take the expectation of e_t on both sides of inequality (29). Then, a new inequality (30) is derived based on (26) and (27), as is shown at the bottom of this paper. It shows that SGD can obtain the expected parameter θ_t as the learning rate is small enough (satisfy $0 < \beta_t < \frac{2}{GB^2}$). At this point, we can get the corresponding state-action function by the obtained parameter θ_t, the effectiveness of SGD is therefore demonstrated. According to the previous description, the convergence of the simplified DMRE is proved theoretically.

B. Feasibility Analysis

To reduce the computation complexity, parameter approximation is introduced for DMRE in this paper, which legitimately simplifies the training targets. However, a potential error m may exist in the simplified DMRE, which is written as:

$$m = Q^i(z,\vec{d}) - \hat{Q}^i(z,\vec{d},\theta) = Q^i(z,\vec{d}) - \sum_{j=1}^{n}\theta_j \phi_{ij}(z). \tag{31}$$

$$\begin{aligned} E[L(\theta_{t+1})] \leq & L(\theta_t) - \beta_t\left(1 - \frac{\beta_t G}{2}\right)\|\nabla L(\theta_t)\|^2 - \beta_t(1 - \beta_t G)\langle \nabla L(\theta_t), E[e_t]\rangle + \frac{\beta_t^2 G\left(B^2 - 1\right)}{2}\|\nabla L(\theta_t)\|^2 \\ \leq & L(\theta_t) - \beta_t\left(1 - \frac{\beta_t G}{2}\right)\|\nabla L(\theta_t)\|^2 + \frac{\beta_t^2 G\left(B^2 - 1\right)}{2}\|\nabla L(\theta_t)\|^2 \\ \leq & L(\theta_t) - \beta_t\left(1 - \frac{\beta_t G B^2}{2}\right)\|\nabla L(\theta_t)\|^2. \end{aligned} \tag{30}$$

According to inequality (30), the target parameter which minimizes the loss function will be obtained via SGD throughout the training process. Thus, the error possibly existing in the simplified DMRE is contingent on the performance of SGD actually. That is to say, the optimal parameter value can minimize the corresponding error. On the other hand, an agent needs to learn N Q-values of all agents and derive strategies from them in the original DMRE, while these Q-values are updated and maintained internally by it. This pattern makes it obvious to get trouble in the curse of dimensionality with the increase of agents. Moreover, the computation complexity increases considerably, as well as the running time. To avoid these bottlenecks, we further give an simplification of DMRE, which legitimately transforms the training targets through parameter approximation in Eq. (14).

References

[1] P. Arthurs, L. Gillam, P. Krause, N. Wang, K. Halder, and A. Mouzakitis, "A Taxonomy and Survey of Edge Cloud Computing for Intelligent Transportation Systems and Connected Vehicles," *IEEE Trans. Intell. Trans. Syst.*, vol. 23, no. 7, pp. 6206–6221, 2022.

[2] K. Jiang, C. Sun, H. Zhou, X. Li, M. Dong, and V. C. M. Leung, "Intelligence-empowered mobile edge computing: Framework, issues, implementation, and outlook," *IEEE Network*, vol. 35, no. 5, pp. 74–82, Oct. 2021.

[3] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource Scheduling in Edge Computing: A Survey," *IEEE Commun. Surveys & Tutorials*, vol. 23, no. 4, pp. 2131-2165, Aug. 2021.

[4] J. Zhang, and K. B. Letaief, "Mobile Edge Intelligence and Computing for the Internet of Vehicles," *Proc. IEEE*, vol. 108, no. 2, pp. 246-261, 2020.

[5] H. Zhou, M. Li, N. Wang, G. Min, and J. Wu, "Accelerating Deep Learning Inference via Model Parallelism and Partial Computation Offloading," *IEEE Trans. Parall. Distr. Syst.*, vol. 34, no. 2, pp. 475-488, 2023.

[6] Y. Zhang, X. Lan, J. Ren, and L. Cai, "Efficient Computing Resource Sharing for Mobile Edge-Cloud Computing Networks," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1227-1240, 2020.

[7] K. Jiang, H. Zhou, D. Zeng, and J. Wu, "Multi-Agent Reinforcement Learning for Cooperative Edge Caching in Internet of Vehicles," *Proc. IEEE Int. Conf. Mobile Ad-Hoc Sensor Syst. (MASS)*, pp. 455-463, 2020.

[8] J. Cui, Y. Liu, Z. Ding, P. Fan, and L. Hanzo, "Next-Generation mm-Wave Small-Cell Networks: Multiple Access, Caching, and Resource Management," *IEEE Veh. Technol. Mag.*, vol. 15, no. 1, pp. 46–53, 2020.

[9] L. Su, and V. K. N. Lau, "Data and Channel-Adaptive Sensor Scheduling for Federated Edge Learning via Over-the-Air Gradient Aggregation," *IEEE Internet of Things J.*, vol. 9, no. 3, pp. 1640-1654, Feb. 2022.

[10] H. Li, K. Ota, and M. Dong, "Deep Reinforcement Scheduling for Mobile Crowdsensing in Fog Computing," *ACM Trans. Internet Techn.*, vol. 19, no. 2, pp. 1-18, 2019.

[11] H. Zhou, Z. Zhang, D. Li, and Z. Su, "Joint Optimization of Computing Offloading and Service Caching in Edge Computing-based Smart Grid," *IEEE Trans. Cloud Comput.*, early access, 2022, doi: 10.1109/TCC.2022.3163750.

[12] J. Ren, N. Cheng, F. Lyu, Q. Ye, and W. Zhuang, "SDN/NFV-Empowered Future IoV With Enhanced Communication, Computing, and Caching," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 274-291, 2020.

[13] W. Jiang, G. Feng, S. Qin, T. Yum, and G. Cao, "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.

[14] H. Zhou, T. Wu, H. Zhang, and J. Wu, "Incentive-driven Deep Reinforcement Learning for Content Caching and D2D Offloading," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2445-2460, 2021.

[15] H. Zhang, M. Huang, H. Zhou, X. Wang, N. Wang, and K. Long, "Capacity Maximization in RIS-UAV Networks: A DDQN-based Trajectory and Phase Shift Optimization Approach," *IEEE Trans. Wireless Commun.*, early access, Sep. 2022, doi: 10.1109/TWC.2022.3212830.

[16] H. Zhang, N. Yang, W. Huangfu, K. Long, and V. C. M. Leung, "Power Control Based on Deep Reinforcement Learning for Spectrum Sharing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 6, pp. 4209-4219, 2020.

[17] H. Zhou, Z. Zhang, Y. Wu, M. Dong, and V. C. M. Leung, "Energy Efficient Joint Computation Offloading and Service Caching for Mobile Edge Computing: A Deep Reinforcement Learning Approach," *IEEE Trans. Green Commun. Netw.*, early access, Jul. 2022, doi: 10.1109/TGCN.2022.3186403.

[18] C. Zhong, M. C. Gursoy, and S. Velipasalar, "Deep Multi-Agent Reinforcement Learning Based Cooperative Edge Caching in Wireless Networks," *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 1-6, May 2019.

[19] Y. Dai, D. Xu, K. Zhang, S. Maharjan, and Y. Zhang, "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.

[20] N. Luong, D. Hoang, S. Gong, D. Niyato, and I. K. Dong, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, May 2019.

[21] X. Ye, M. Li, P. Si, R. Yang, Z. Wang, and Y. Zhang, "Collaborative and Intelligent Resource Optimization for Computing and Caching in IoV With Blockchain and MEC Using A3C Approach," *IEEE Trans. Veh. Technol.*, early access, 2022, doi: 10.1109/TVT.2022.3210570.

[22] B. Jedari, G. Premsankar, G. Illahi, M. D. Francesco, A. Mehrabi, and A. Ylä-Jääski, "Video Caching, Analytics, and Delivery at the Wireless Edge: A Survey and Future Directions," *IEEE Commun. Surveys & Tutorials*, vol. 23, no. 1, pp. 431–471, 2021.

[23] H. Wu, and H. Lu, "Delay and Power Tradeoff With Consideration of Caching Capabilities in Dense Wireless Networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 5011–5025, Oct. 2019.

[24] M. Yan, C. A. Chan, W. Li, L. Lei, and I. Chih-Lin, "Assessing the Energy Consumption of Proactive Mobile Edge Caching in Wireless Networks," *IEEE Access*, vol. 7, pp. 104394-104404, Jul. 2019.

[25] X. Zhang, Z. Qi, G. Min, W. Miao, Q. Fan, and Z. Ma, "Cooperative Edge Caching Based on Temporal Convolutional Networks," *IEEE Trans. Parall. Distr. Syst.*, vol. 33, no. 9, pp. 2093-2105, Sep. 2022.

[26] K. Zhang, J. Cao, S. Maharjan, and Y. Zhang, "Digital Twin Empowered Content Caching in Social-Aware Vehicular Edge Networks," *IEEE Trans. Computat. Social Syst.*, vol. 9, no. 1, pp. 239-251, Feb. 2022.

[27] J. Zhao, X. Sun, Q. Li, and X. Ma, "Edge Caching and Computation Management for Real-Time Internet of Vehicles: An Online and Distributed Approach," *IEEE Trans. Intell. Trans. Syst.*, vol. 16, no. 10, pp. 2183-2197, April 2021.

[28] W. C. Ao, and K. Psounis, "Fast Content Delivery via Distributed Caching and Small Cell Cooperation," *IEEE Trans. Mobile Comput.*, vol. 17, no. 5, pp. 1048-1061, May 2018.

[29] Z. Gu, H. Lu, P. Hong, and Y. Zhang, "Reliability Enhancement for VR Delivery in Mobile-Edge Empowered Dual-Connectivity Sub-6 GHz and mmWave HetNets," *IEEE Trans. Wireless Commun.*, vol. 21, no. 4, pp. 2210-2226, April 2022.

[30] S. Zhang, and J. Liu, "Optimal Probabilistic Caching in Heterogeneous IoT Networks," *IEEE Internet of Things J.*, vol. 7, no. 4, pp. 3404–3414, Apr. 2020.

[31] R. Wang, Z. Kan, Y. Cui, D. Wu, and Y. Zhen, "Cooperative Caching Strategy With Content Request Prediction in Internet of Vehicles," *IEEE Internet of Things J.*, vol. 8, no. 11, pp. 8964-8975, Jun. 2021.

[32] Y. Dai, D. Xu, S. Maharjan, G. Qiao, and Y. Zhang, "Artificial intelligence empowered edge computing and caching for internet of vehicles," *IEEE Wireless Commun. Mag.*, vol. 26, no. 3, pp. 12-18, Jun. 2019.

[33] L. T. Tan, and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10190-10203, Jul. 2018.

[34] G. Qiao, S. Leng, S. Maharjan, Y. Zhang, and N. Ansari, "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.

[35] X. Wang, C. Wang, X. Li, V. C. M. Leung, and T. Taleb, "Federated Deep Reinforcement Learning for Internet of Things With Decentralized Cooperative Edge Caching," *IEEE Internet of Things J.*, vol. 7, no. 10, pp. 9441-9455, Oct. 2020.

[36] H. Tian, et al., "CoPace: Edge Computation Offloading and Caching for Self-Driving With Deep Reinforcement Learning," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 13281-13293, Dec. 2021.

[37] X. Li, X. Wang, P. Wan, Z. Han, and V. C. M. Leung, "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.

[38] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE Trans. Netw.*, vol. 24, no. 5, pp. 2795-2808, Oct. 2016.

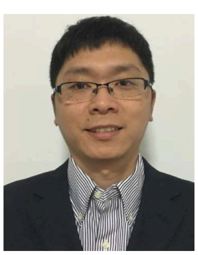
[39] J. Hu, and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *J. Mach. Learn. Res.*, vol. 4, pp. 1039–1069, 2003.

[40] P. Casgrain, B. Ning, and S. Jaimungal, "Deep Q-learning for Nash equilibria: Nash-DQN," *arXiv:1904.10554. [Online]*, May 2019. Available: http://arxiv.org/abs/1904.10554.

[41] Z. Zhang, D. Wang, and J. Gao, "Learning Automata-Based Multiagent Reinforcement Learning for Optimization of Cooperative Tasks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 10, pp. 4639-4652, Oct. 2021.

Huan Zhou (M'14) received his Ph. D. degree from the Department of Control Science and Engineering at Zhejiang University. He was a visiting scholar at the Temple University from Nov. 2012 to May, 2013, and a CSC supported postdoc fellow at the University of British Columbia from Nov. 2016 to Nov. 2017. Currently, he is a professor at the College of Computer, Northwestern Polytechnical University. He was a Lead Guest Editor of Pervasive and Mobile Computing, and TPC Chair of EAI BDTA 2020, Local Arrangement Chair of I-SPAN 2018, and TPC member of IEEE MASS, ICCCN, Globecom, ICC, WCSP, etc. He has published more than 70 research papers in some international journals and conferences, including IEEE JSAC, TPDS, TMC, TWC and so on. His research interests include Mobile Social Networks, VANETs, Opportunistic Mobile Networks, and mobile data offloading. He receives the Best Paper Award of I-SPAN 2014 and I-SPAN 2018, and is currently serving as associate editor for EURASIP Journal on Wireless Communications and Networking and IEEE ACCESS.

Kai Jiang (S'20) is currently pursuing the Ph. D. degree in cyberspace security at Wuhan University, China. His research interests include Mobile Edge Computing, Deep Reinforcement Learning, and Internet of Vehicles.



Shibo He (SM'19) is currently a Full Professor with Zhejiang University and also with Alibaba-Zhejiang University Joint Research Institute of Frontier Technologies. His research interests include the Internet of Things, Crowdsensing, and big data.

Dr. He serves on the Editorial Board for several journals including IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. He served as a TPC Co-Chair for IEEE ScalCom 2014, Finance & Registration Chair for ACM MobiHoc 2015, and a Symposium Co-Chair for IEEE ICC 2017 and IEEE Globecom 2020.

Geyong Min is a Professor of High Performance Computing and Networking in the Department of Computer Science within the College of Engineering, Mathematics and Physical Sciences at the University of Exeter, United Kingdom. He received the PhD degree in Computing Science from the University of Glasgow, United Kingdom, in 2003, and the B.Sc. degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. His research interests include Computer Networks, Wireless Communications, Parallel and Distributed Computing, Ubiquitous Computing, Multimedia Systems, Modelling and Performance Engineering.

Jie Wu (F'09) is the Chair and a Laura H. Carnell Professor in the Department of Computer and Information Sciences at Temple University, Philadelphia, PA, USA. Prior to joining Temple University, he was a Program Director at the National Science Foundation and a Distinguished Professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. Dr. Wu has regularly published in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE Transactions on Services Computing, and the Journal of Parallel and Distributed Computing. Dr. Wu was general co-chair/chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, ACM MobiHoc 2014, IEEE ICPP 2016, and IEEE CNS 2016, as well as program co-chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a CCF Distinguished Speaker and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.