Towards Cooperative Caching for Vehicular Networks with Multi-level Federated Reinforcement Learning

Lei Zhao[1], Yongyi Ran*[2], Hao Wang[1], Junxia Wang[1], Jiangtao Luo*[2]
Chongqing University of Posts and Telecommunications, Chongqing, China
[1]{S190131061, S190101069, D170101010}@stu.cqupt.edu.cn, [2]{Ranyy, Luojt}@cqupt.edu.cn

ICC 2021 - IEEE International Conference on Communications | 978-1-7281-7122-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICC42927.2021.9500714

***Abstract*—Content caching in vehicular networks is a promising technology to dramatically reduce the request-response time and transmission delay. The existing caching policies often suffer from high computation and communication overhead and fail to well capture the dynamics of the vehicular networks and content popularity. In this paper, we propose a novel *Co*operative *Ca*ching algorithm for vehicular networks with multi-level federated *R*einforcement *L*earning (named CoCaRL) to dynamically determine which contents should be replaced and where the content requests should be served. In CoCaRL, Deep Reinforcement Learning (DRL) is employed to optimize the cooperative caching policy between RoadSide Units (RSUs) of vehicular networks, while a federated learning framework applies to reduce the computation and communication overhead in a decentralized way. To speed-up the convergence rate, we also develop a two-level aggregation mechanism for federated learning, where the low-level aggregation is performed at the RSUs and the high-level aggregation is executed at a Global Aggregator (GA). Through extensive simulation experiments, we demonstrate that our algorithm can: 1) achieve a higher hit rate than four baseline algorithms, 2) converge faster than original federated reinforcement learning without multi-level aggregation, and 3) perform good adaptability to different cache capacities and content quantities.**

***Index Terms*—Vehicular Networks, Cooperative Caching, Deep Reinforcement Learning, Federated Learning**

I. Introduction

Along with the recent advances in automotive technology and in-vehicle networks, a great number of innovative vehicular applications are emerged to enhance driving safety, travel comfort, and in-car entertainment [1]. These applications often impose a large demand on computation, communication, and storage resources, and have specific requirements on Quality of Service (QoS) (e.g., transmission delay and response time), which cannot be satisfied if the data is fetched only from a cloud data center. Vehicular edge caching is considered as a promising technology to fulfill such QoS requirements by partially migrating the cloud caching to edge vehicular devices, such as RoadSide Units (RSUs) [2]. However, due to limited cache capacity at RSUs, a caching scheme by cooperatively utilizing the cache space of the cloud data center and all the RSUs should be well designed.

*The corresponding authors are Yongyi Ran and Jiangtao Luo.

*This work is supported by National Science Foundation of China (No. 62003067).

Besides the limited cache capacity of RSUs, the design of caching policy for vehicular networks will also face the following three issues. **First**, the regional content popularity and validity is time-varying. The cache policy needs to dynamically decide which contents need to be cached and where the selected contents are cached (e.g., local RSU or neighbor RSU) by making full use of the dynamics of cache space and content popularity. **Second**, there exist a large number of connected vehicles and contents, which will result in high computation and communication costs as well as the problem of "Curse of Dimensionality". **Third**, privacy rules limit the use of user data. In order to protect the privacy of vehicle users, the user data (e.g., request interests, procurement records) sometimes can only be used locally and cannot be transmitted to a centralized place. It makes centralized algorithms impossible to solve a global optimal caching policy.

Many caching policies have been designed and applied in vehicular networks. Traditional caching approaches, such as Least Recently Used (LRU), Least Frequently Used (LFU), and First Input First Output (FIFO), is firstly employed to derive the basic caching replacement policies in vehicular scenarios [3], [4], while they often fail to well capture the dynamics of content popularity. In [5], [6], the content popularity is considered to find a content placement strategy that decides the placing locations and proportions for all contents, but the formulated problems are often NP-hard and only a near-optimal solution can be derived due to high computation complexity. To handle this issue, some model-free caching algorithms based on deep reinforcement learning (DRL) are proposed in vehicular networks [7]–[10]. However, these algorithms need to collect a lot of data for training at a centralized place, which may result in privacy issues and make the convergence rate slow. To alleviate this, [11] proposes a novel decentralized caching scheme based on federated DRL for the Internet of things, but only the devices covered by the same RSU will federally share the training model, and the global content popularity cannot be perceived.

To tackle the challenges discussed above, we propose a collaborative caching algorithm for vehicular networks combining DRL with multi-level federated learning, called CoCaRL. In CoCaRL, vehicles collect and keep local data individually, then train a DRL model separately on their own On-board

Unit (OBU), where the initial parameters of the DRL model on each OBU are the same. After one round of training, the new model parameters on each OBU are sent to the associated RSU, where all the model parameters from participating vehicles are aggregated, here we call this procedure as **Low-level Aggregation**. All the RSUs then send the aggregated parameters to a central server, named Global Aggregator (GA), to complete a **High-level Aggregation**. Finally, the GA will send the aggregated parameters to each OBU on vehicles for the next round of training. In this way, **GA can help the training models on the OBUs to perceive the changes in global content popularity**. Our main contributions are summarized as follows:

- We propose the CoCaRL algorithm combining DRL with federated learning to optimize the caching policy for vehicular networks. DRL can well capture the dynamics of content popularity and cache capacity, while federated learning can reduce the computation and communication overhead by training a model in a decentralized way.
- In order to speed-up the convergence rate, we propose a multi-level aggregation mechanism for federated learning. The low-level aggregation is performed at RSU, while the high-level aggregation is finished at the GA.
- Extensive simulation experiments are carried out to demonstrate that the proposed CoCaRL can improve the convergence rate and cache hit rate compared to the baseline algorithms.

The rest of the paper is organized as follows. Section II summarizes the related work. The system model and problem formulation is presented in Section III. The proposed CoCaRL algorithm is described in Section IV. Section V shows the evaluation results. Finally, section VI concludes this paper.

II. Related Work

In this section, we investigate the relevant work from two aspects: traditional methods and learning-based methods for vehicular caching.

A. Traditional methods for vehicular caching

Many efforts based on traditional methods have been devoted to addressing the caching problem of vehicular networks. Su *et al.* [12] proposed a model to determine whether and where to obtain the replica of content when the moving vehicle requests it, then a cross-entropy-based dynamic content caching scheme is proposed accordingly to cache the contents at the edge of VCNs. In [13], the authors aimed at minimizing the average service delay. Lyapunov optimization and matching theory are combined to optimize the cache replacement in the vehicle. Chen *et al.* [5] formulated an optimization problem of cooperative content placement to minimize the overall transmission delay and service cost. An ant colony optimization-based algorithm is developed to find a near-optimal solution. Liu *et al.* [6] jointly optimized the cache allocation and content placement in vehicular networks, and developed a low-complexity approximate algorithm which performs within a bounded gap to the optimum. These traditional methods often require global information and are difficult to adapt to dynamic environments. In addition, the formulated problems are often NP-hard and only a near-optimal solution can be derived due to high computation complexity.

B. Learning-based methods for vehicular caching

With the development of machine learning/deep learning, many learning-based methods have been developed to optimize the content caching strategy for vehicular networks. In [8], the authors described vehicular content caching as a long-term mixed-integer linear programming problem and built a collaborative caching scheme based on Deep Deterministic Policy Gradient (DDPG). Ma *et al.* [9] modeled the pre-caching and task allocation as Markov decision processes (MDP), and DDPG is applied to determine the optimal ratio of pre-caching and task allocation. Yu *et al.* [10] deployed a context-aware adversarial auto-encoder model to estimate content popularity, and the predicted popular contents are placed at the edge of vehicular networks to reduce latency. Zhong *et al.* [7] studied content caching at the wireless network edge using deep actor-critic reinforcement learning with Wolpertinger architecture. Most of the learning-based caching schemes are performed in a centralized way, the computation and communication overhead is extremely large, and the convergence rate is slow. In addition, there exist some privacy issues due to gathering the training data in a central place.

III. System Model And Problem Formulation

In this section, we first describe the vehicle network model. Second, we present the content model. Finally, the problem formulation is given.

A. System Model for Vehicular Networks

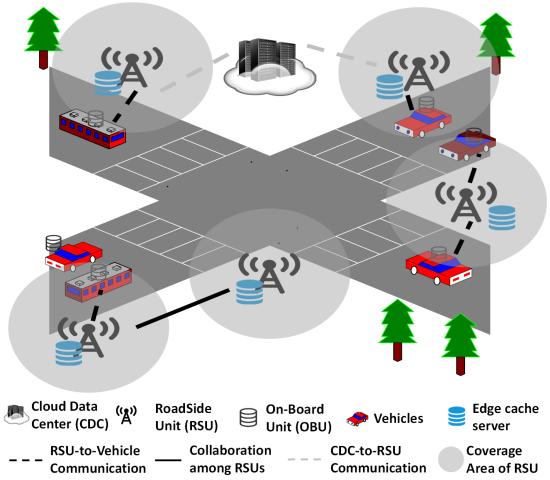

Fig. 1. System model of vehicular networks for content delivering.

As shown in Fig. 1, a typical vehicular network for content delivering comprises one Cloud Data Center (CDC), a set of RoadSite Units (RSUs) $\mathcal{U} = \{1, 2, \ldots, U\}$, and a number of

vehicles $\mathcal{V} = \{1, 2, \ldots, V\}$. Each vehicle is equipped with an On-Board Unit (OBU) and the set of OBUs can be represented as $\mathcal{O} = \{1, 2, \ldots, O\}$. CDC has enough cache space and can store all published contents. RSUs provide content services for vehicles within the coverage area through wireless links, and the adjacent RSUs communicate with each other via wired optical cables. RSUs have the limited capability of caching and computing, and can selectively cache part of the contents. Here we assume that the cache size of RSU $u \in \mathcal{U}$ is S_u. In addition, RSUs will be used to aggregate and store model parameters while training our proposed algorithm. Vehicle users can send requests via its OBU to obtain content from the associated RSU, neighboring RSUs, or the CDC. Model training is separately carried out on each OBU of the vehicles.

B. Content Model

We define the content library as $\mathcal{C} = \{1, 2, \ldots, C\}$, and denote B_c as the size of content $c \in \mathcal{C}$.

Content popularity: Let $p_{u,i}$ denotes the popularity of the i-th contents at RSU u, which follows the Mandelbrot-Zipf (MZipf) distribution [14]. Thus we can get

$$p_{u,i} = \frac{I_u(i)^{-z_u}}{\sum_{u=1}^{|\mathcal{U}|} I_u(i)^{-z_u}}, \tag{1}$$

where $I_u(i)$ indicates the popularity rank of the i-th content at RSU u, z_u is a skewness factor taking values in $[0.6, 1.2]$. A larger value for z_u indicates that RSU u contains relatively a small set of very popular contents.

C. Problem Formulation

In order to adapt to the changes in the environment, the RSUs need to dynamically determine which contents should be replaced and where the content requests should be served during each episode. Through such a decision, our optimization goal is to maximize the cache hit rate. The cache hit rate P_{hit} of T requests can be defined as

$$P_{hit} = \frac{\sum_{t=1}^{T} H\left(index\right)}{T} \tag{2}$$

where the function $H(index)$ is defined as:

$$H\left(index\right) = \begin{cases} 1, & \text{content request } index \text{ hit} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Therefore, the problem of maximizing the cache hit rate can be expressed as:

$$\begin{aligned} \max_{\boldsymbol{\phi}} \quad & P_{hit} \\ \text{s.t.} \quad & \sum_{c=1}^{|\mathcal{C}|} \phi_c B_c \leq S_u, \forall u \in \mathcal{U} \end{aligned} \tag{4}$$

where $\boldsymbol{\phi}$ is a $1 \times |\mathcal{C}|$ matrix, representing the content cached in the RSU, $\phi_c = 1$ means RSU has a cache of content c, $\phi_c = 0$ is the opposite.

IV. The Proposed CoCaRL Algorithm

In this section, we first give some basics of DRL framework, and then describe the proposed CoCaRL algorithm in detail.

A. The DRL Model

Here we define the state space, action space and reward function for the DRL-based framework as follows:

*1) **State space**:* We divide the state into three parts: content request state $\boldsymbol{s}_o^{re} = \{s_{o,c}^{re}\}$, RSU content cache state $\boldsymbol{s}_u^{ca} = \{s_{u,c}^{ca}\}$ and content popularity state $\boldsymbol{s}_u^{po} = \{s_{u,c}^{po}\}$, $o \in \mathcal{O}$, $u \in \mathcal{U}$, $c \in \mathcal{C}$. At the beginning of each decision episode, we will collect all the states for training or decision. $s_{o,c}^{re} = 1$ indicates OBU o sends a request for content c, $s_{o,c}^{re} = 0$ represents that OBU o will not request content c during this episode. $s_{u,c}^{ca} = 1$ denotes RSU u cached content c, $s_{u,c}^{ca} = 0$ is the opposite. $s_{u,c}^{po} = p_{u,c}$ means the content popularity of content c under RSU u. The state vector can be expressed as

$$\mathbf{S} = (\boldsymbol{s}_o^{re}, \boldsymbol{s}_u^{ca}, \boldsymbol{s}_u^{po}) \tag{5}$$

*2) **Action space**:* In order to adapt to the dynamics of the system states and guarantee the QoS, the RSUs need to determine which contents should be replaced and where the content requests should be served during each episode. Here we define three types of actions following [11]: local actions $\boldsymbol{a}^{loc} = \{a_c^{loc}\}$, collaboration actions $\boldsymbol{a}^{co-rsu} = \{a_u^{co-rsu}\}$, and cloud action a^{cdc}, where $a_c^{loc}, a_u^{co-rsu}, a^{cdc} \in \{0, 1\}$, $u \in \mathcal{U}$, $c \in \mathcal{C}$. Therefore, the action vector can be represented as

$$\mathbf{A} = (\boldsymbol{a}^{loc}, \boldsymbol{a}^{co-rsu}, a^{cdc}) \tag{6}$$

For the local actions, $a_c^{loc} = 1$ represents content c needs to be replaced by the current request content, $a_c^{loc} = 0$ is the opposite. For the cooperation actions, $a_u^{co-rsu} = 1$ indicates the current content request is processed by the RSU u. For the cloud action, $a^{cdc} = 1$ means the requested content should be downloaded from CDC.

*3) **Reward function**:* According to the request processing procedure discussed above, the content may be hit in the local RSU, the neighbor RSU, or the CDC. The hit in different locations will get different response times and transmission delay. Therefore, our reward is correspondingly divided into three cases: local RSU hit rate P_{hit}^{local}, neighbor RSU hit rate P_{hit}^{co-rsu} and CDC hit rate. Accordingly, the reward function can be defined as follows:

$$r\left(\mathbf{S}, \mathbf{A}\right) = \begin{cases} P_{hit}^{local}, & \text{Local} \\ P_{hit}^{co-rsu}, & \text{CO-RSU} \\ 0, & \text{CDC} \end{cases} \tag{7}$$

If one OBU obtains the requested content from local RSU, the reward is P_{hit}^{local}. If the OBU get the requested content from a neighbor RSU, the reward is P_{hit}^{co-rsu}. If the OBU downloads the requested content from the CDC, the reward is set as 0.

B. CoCaRL: Cooperative Content Caching with Multi-level Federated RL Algorithm

Although some progress has been achieved in DRL-based caching policy for vehicular networks (as discussed in Section II), most of the DRL-based approaches have the following shortcomings: 1) The convergence rate is slow if only using

one DRL agent for training, 2) the computation and communication overhead is large if training in a centralized way, and 3) user privacy cannot be guaranteed if using user data at a central place. To address these issues, we propose a cooperative content caching algorithm by combining DRL with federated learning, termed CoCaRL. Especially, we develop a multi-level aggregation mechanism for CoCaRL to speed-up training convergence and make the local agents perceive the changes in global content popularity.

*1) **Individual DRL Model on each OBU**:* For each OBU, we implement an DRL agent for individually training a Double Deep Q-Network (DDQN) [15] model with local state. DDQN is extended from Q-learning by utilizing neural network. In Q-learning, Q-value function is defined as the expected cumulative reward to evaluate how good the chosen action is for its corresponding state:

$$Q(\mathbf{S},\ \mathbf{A}) = E[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{S}_t, \mathbf{A}_t) | \mathbf{S}_0 = \mathbf{S}, \mathbf{A}_0 = \mathbf{A}] \quad (8)$$

where γ is a discount factor. According to Bellman equation, the Q-value function can be expressed as:

$$Q(\mathbf{S}, \mathbf{A}) = R(\mathbf{S}, \mathbf{A}) + \gamma \arg\max_{A' \in \mathcal{A}} Q(\mathbf{S}', \mathbf{A}') \quad (9)$$

Once the $Q^*(\mathbf{S}, \mathbf{A})$ for each state-action pair is obtained, the optimal policy π^* can be expressed as:

$$\pi^*(\mathbf{S}) = \arg\max_{A \in \boldsymbol{\mathcal{A}}} Q^*(\mathbf{S}, \mathbf{A}) \quad (10)$$

DDQN uses neural network to approximate the Q-value. The DRL agent preserves two neural networks $Q(\mathbf{S}, \mathbf{A}; \boldsymbol{w})$ and $\hat{Q}(\mathbf{S}, \mathbf{A}; \hat{\boldsymbol{w}})$, $\boldsymbol{w}$ and $\hat{\boldsymbol{w}}$ are the network parameters. The Q network is an evaluation network for selecting an action, while the $\hat{Q}$ network is a target network for training. The DDQN model on each OBU is initialized with the same parameters. Each DRL agent first executes the corresponding action $\mathbf{A}(\mathbf{S})$ according to the current local state $\mathbf{S}$, and then obtains the current feedback reward $r(\mathbf{S}, \mathbf{A}(\mathbf{S}))$. Finally, the previous system state $\mathbf{S}$ is transformed into the next new system state $\mathbf{S}'$. The transition $< \mathbf{S}, \mathbf{A}(\mathbf{S}), r(\mathbf{S}, \mathbf{A}(\mathbf{S})), \mathbf{S}' >$ will be stored into an experience replay pool M (namely, transition memory). For each episode $e > 0$, the agent randomly selects a mini-batch m_e from the transition memory M, and then trains the Q network by minimizing the loss function:

$$\begin{aligned} L(\boldsymbol{w}^e) = \mathbb{E}_{(\mathbf{S},\mathbf{A},r(\mathbf{S},\mathbf{A}),\mathbf{S}') \in m^e}[(r(\mathbf{S}, \mathbf{A}) \\ + \gamma \cdot \hat{Q}(\mathbf{S}, \arg\max_{\mathbf{A}'} Q(\mathbf{S}', \mathbf{A}'; \boldsymbol{w}^e); \hat{\boldsymbol{w}}^e) \\ - Q(\mathbf{S}, \mathbf{A}; \boldsymbol{w}^e)^2] \end{aligned} \quad (11)$$

We can obtain the gradient updates of w^e by $\nabla_{w^e} L(w^e)$ as follows:

$$\begin{aligned} \nabla_{w^e} L(w^e) = \mathbb{E}_{(S,A,r(S,A),S') \in m^e}[(r(S, A) \\ + \gamma \cdot \hat{Q}\left(S, \arg\max_{A'} Q(S', A'; w^e); \hat{w}^e\right) \\ - Q(S, A; w^e)) \cdot \nabla_{w^e} Q(S, A; w^e)] \end{aligned} \quad (12)$$

After each round of training, OBU o updates the parameters through the following formula:

$$w_o^{e+1} = w_o^e - \eta \nabla L_o(w^e), \quad (13)$$

where η represents the learning rate.

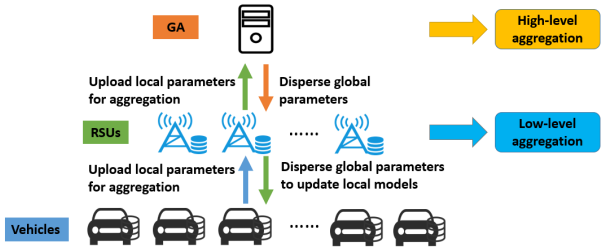

Fig. 2. Illustration of low-level and high-level aggregation of CoCaRL.

*2) **Multi-level Federated DRL**:* As shown in Fig. 2, our proposed CoCaRL includes two levels of aggregation: Low-level aggregation is performed in RSUs, while high-level aggregation is executed in the Global Aggregator (GA). GA is essentially a centralized model parameter server, which can be located at a standalone server or implemented on any RSU.

Algorithm 1 The training of CoCaRL

Initialization:
Initial weight w_0;
The number of episodes E;
Learning Rate η.
Initial replay memory $\mathcal{M}$;
Initial Q network and $\hat{Q}$ with same weight w.
Iteration:
for $e = 1, 2, 3 \ldots, E$ **do**
  for $u = 1, 2, 3 \ldots, U$ **do**
    for each OBU h **do**
      Receive state $\mathbf{S}_e$.
      Select action $\arg\max_{\mathbf{A}(\mathbf{S}_e)} Q(\mathbf{S}_e, \mathbf{A}(\mathbf{S}_e); \boldsymbol{w}^e)$.
      Execute action $\mathbf{A}_e$
      Obtain reward $r(\mathbf{S}_e, \mathbf{A}_e)$.
      new state $\mathbf{S}_{e+1}$.
      Store transition ($\mathbf{S}_e$,$\mathbf{A}_e$,r,$\mathbf{S}_{e+1}$) in $\mathcal{M}$.
      Sample a random minibach of m_e from $\mathcal{M}$.
      Compute the loss by (11)
      Update the Q network parameters w_h^{e+1} according to (13)
      Upload w_h^{e+1} to RSU u.
    end for
    Receive w_h^{e+1} from each OBS h
    Update local RSU parameter by (14)
    Upload w_u^{e+1} to GA.
  end for
  Receive w_u^{e+1} from each RSU u
  Update global parameter according to (15)
  GA distributes w^{e+1} to each OBU.
end for

Low-level aggregation: This aggregation occurs between the RSU and its covered OBUs. At the beginning of training, the DRL model on each OBU is initialized with the same network parameters. After each round of training, the OBU uploads the new model parameters to its associated RSU, where all the model parameters from participating vehicles are aggregated. Assume that there are H OBUs in the coverage of RSU u. The datasets of H OBUs can be represented

as $\mathcal{D}_u = \{D_1, D_2, \ldots D_H\}$. The low-level aggregation that occurs on RSU u is denoted as:

$$w_u^{e+1} = \frac{\sum_{h=1}^{H} D_h w_h^{e+1}}{\sum_{h=1}^{H} D_h} \tag{14}$$

In this process, the model training and data storing are all locally performed on the OBUs, which means that the computing cost and power consumption are shared by all the OBUs. In addition, there is no exchange of native data between OBUs and RSUs, the privacy can be well protected.

High-level aggregation: This aggregation occurs between GA and RSUs. After all the RSUs aggregates the parameters collected from its covered OBUs, RSUs upload the aggregated parameters to the GA. Then, GA updates the global parameters and sends the updated parameters to each OBU via RSU for the next round of training. This high-level aggregation formula is defined as:

$$w^{e+1} = \frac{\sum_{u=1}^{|\mathcal{U}|} w_u^{e+1}}{|\mathcal{U}|} \tag{15}$$

In this way, GA can help the training models on the OBUs to perceive the changes of global content popularity, and speed-up the convergence of the models. The pseudo code of training CoCaRL is shown in Algorithm 1.

V. Performance Evaluation

In this section, we present simulation experiments to evaluate our proposed algorithm, and analyze the simulation results.

A. Experiment Setup

In order to evaluate our proposed algorithm, we use Python to simulate a vehicular network environment in an urban area. The environment consists of a CDC, 2 RSUs, and several vehicles located in the RSU coverage area. Most of the existing methods use Zipf distribution to simulate the distribution of content popularity, therefore, we use Zipf distribution to generate a dataset. All vehicles have datasets for local model training. RSU capacity S = 400 MB, content type C = 100. The size of each content is [1,8] MB. The hidden layer of the Q network has 256 neurons, reward discount factor γ = 0.9, learning rate η = 0.05. The capacity of replay memory M = 2000, the size of minibatch = 100.

B. Baseline Algorithms and Metrics

We compare our proposed CoCaRL algorithm with four baseline solutions::

- Least Recently Used (LRU): The least recently used content will be replaced firstly.
- Least Frequently Used (LFU): The least frequently used content will be replaced firstly.
- First Input First Output (FIFO): The oldest content will be replaced firstly.
- Federated DRL (FDRL): A dynamic cache control algorithm based on federated DRL [11].

The metrics used in this paper to evaluate the performance includes:

1) Hit rate: The hit rate is calculated follows the formula (2).
2) Transmission delay: The transmission delay includes three parts: RSU to vehicle t_{rv}, RSU to RSU t_{rr} and RSU to CDC t_{rc}.

C. Convergence Results & Performance Comparison

In Fig. 3, we can observe that the convergence time of CoCaRL is about 1000 episodes earlier than that of FDRL. This is because CoCaRL algorithm using multi-level federation can better aggregate the popularity of each RSU cache content than FDRL. Besides, CoCaRL is adaptable to dynamic environments and makes good decisions in complex environments.

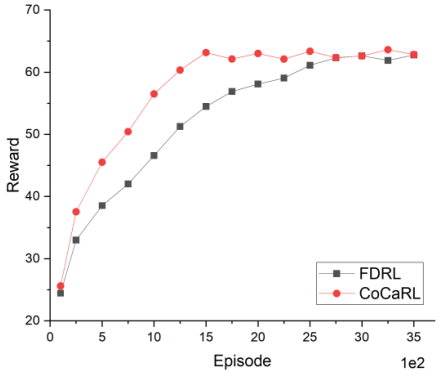

Fig. 3. Comparison of the convergence speed of CoCaRL and FDRL.

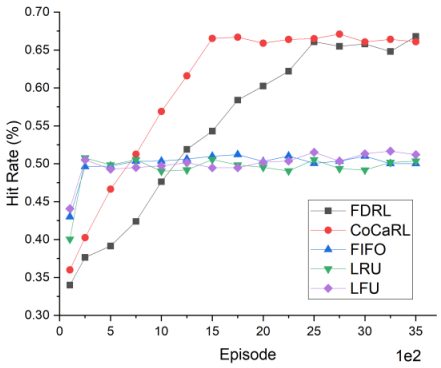

Fig. 4. Hit rate of different algorithms.

As shown in Fig. 4 and Fig. 5, the three traditional algorithms FIFO, LRU and LFU, cannot achieve high hit rate and low delay in a complex environment, CoCaRL and FDRL outperform traditional algorithms. Compared with FIFO, LRU and LFU algorithms, its hit rate performance has improved by 15%, 15% and 10%, and its latency performance has increased by 28%, 27% and 24%, respectively. In the initial stage, the performance of CoCaRL and FDRL is worse than that of traditional algorithms. This is because learning-based algorithms require a certain amount of time to interact with the environment and continue to learn while traditional algorithms have fixed mechanisms and cannot make flexible decisions based on dynamic environments. But in the later stage, the

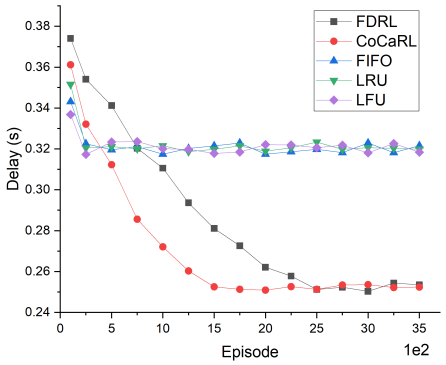

Fig. 5. Delay of different algorithms.

hit rate of CoCaRL and FDRL is significantly better than traditional algorithms. Besides, in many cases, our algorithm is better than FDRL because we have high-level aggregation that allows vehicles to perceive global popularity.

D. Sensitivity Analysis

Fig. 6 demonstrates the algorithm hit rate under different cache capacities. The cache capacity ranges from 100 to 400. Compared with the FIFO, LRU and LFU algorithms, the CoCaRL algorithm improves the hit rate performance by 11%, 9%, and 9% respectively and slightly higher than the FDRL. We can see from the figure that as the cache capacity increases, the cache hit rate of various algorithms is increasing because more cache capacity means more popular content can be stored, and the process of cache replacement rarely occurs.

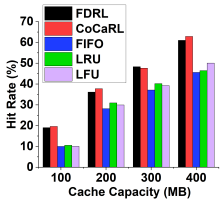

Fig. 6. Hit rate with different cache capacities.

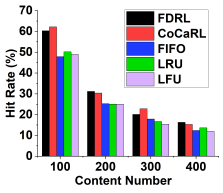

Fig. 7. Hit rate with different content numbers

We compare the cache hit rate of the algorithm under different content amounts in Fig. 7. The CoCaRL algorithm improves the hit rate performance with up to 7%, 6% and 7% compared to the FIFO, LRU and LFU algorithms, respectively. As the number of content increases, the cache hit rate of various algorithms decreases significantly. Because the cache capacity is limited, some moderately popular content may not be cached, resulting in frequent cache replacement processes.

VI. Conclusion

In this paper, we have proposed a Cooperative Content Caching algorithm with Multi-level Federated Reinforcement Learning, named CoCaRL, to dynamically determine which contents should be replaced and where the content requests should be served. In CoCaRL, each OBU firstly trains a DRL model individually and feedbacks the model parameters to its associated RSU. Then, a multi-level aggregation mechanism is performed to achieve federated training and accelerate the overall convergence speed. The low-level aggregation is performed at the RSU, while the high-level aggregation is finished at a Global Aggregator. Finally, extensive simulation experiments are carried out to demonstrate that our proposed CoCaRL can: 1) achieve a higher hit rate than four baseline algorithms, 2) converge faster than original federated reinforcement learning without multi-level aggregation, and 3) perform good adaptability to different cache capacities and content quantities.

References

[1] J. Zhang and K. B. Letaief, "Mobile edge intelligence and computing for the internet of vehicles," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 246–261, 2020.

[2] Y. Wang, S. He, X. Fan, C. Xu, and X. Sun, "On cost-driven collaborative data caching: A new model approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 3, pp. 662–676, 2019.

[3] W. Zhao, Y. Qin, D. Gao, C. H. Foh, and H. Chao, "An efficient cache strategy in information centric networking vehicle-to-vehicle scenario," *IEEE Access*, vol. 5, pp. 12 657–12 667, 2017.

[4] H. Khelifi, S. Luo, B. Nour, and H. Moungla, "In-network caching in icn-based vehicular networks: Effectiveness performance evaluation," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[5] J. Chen, H. Wu, P. Yang, F. Lyu, and X. Shen, "Cooperative edge caching with location-based and popular contents for vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 10 291–10 305, 2020.

[6] T. Liu, S. Zhou, and Z. Niu, "Joint optimization of cache allocation and content placement in urban vehicular networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[7] C. Zhong, M. C. Gursoy, and S. Velipasalar, "Deep reinforcement learning-based edge caching in wireless networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 1, pp. 48–61, 2020.

[8] 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 Journal*, vol. 7, no. 1, pp. 247–257, 2020.

[9] T. Ma, X. Chen, Z. Ma, and Y. Chen, "Deep reinforcement learning for pre-caching and task allocation in internet of vehicles," in *2020 IEEE International Conference on Smart Internet of Things (SmartIoT)*, 2020, pp. 79–85.

[10] Z. Yu, J. Hu, G. Min, Z. Zhao, W. Miao, and M. S. Hossain, "Mobility-aware proactive edge caching for connected vehicles using federated learning," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 2020.

[11] 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 Journal*, vol. 7, no. 10, pp. 9441–9455, 2020.

[12] Z. Su, Y. Hui, Q. Xu, T. Yang, J. Liu, and Y. Jia, "An edge caching scheme to distribute content in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 6, pp. 5346–5356, 2018.

[13] 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 Transactions on Intelligent Transportation Systems*, pp. 1–15, 2020.

[14] H. Wu, J. Li, H. Lu, and P. Hong, "A two-layer caching model for content delivery services in satellite-terrestrial networks," in *2016 IEEE Global Communications Conference (GLOBECOM)*, 2016, pp. 1–6.

[15] H. V. . Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *2016 AAAI*, 2016, pp. 2094–2100.