Multi-Timescale Hierarchical Prefetching for Online Caching in Vehicular Edge Networks

Shuaibing Lu[*‡], Bojin Xiang[*], Jie Wu[†], Philipp Andelfinger[‡], and Wentong Cai[‡]
[*]College of Computer Science, Beijing University of Technology, Beijing, China
lushuaibing@bjut.edu.cn, sean-bonjean@emails.bjut.edu.cn
[†]Temple University, Philadelphia, USA, jiewu@temple.edu
[‡]Nanyang Technological University, Singapore, {philipp.andelfinger, aswtcai}@ntu.edu.sg

***Abstract*—Content delivery in vehicular edge networks faces critical challenges due to dynamic user mobility, unpredictable content request patterns, and limited storage at edge nodes. To tackle these problems, we propose a distributed online framework that jointly performs proactive caching at roadside units (RSUs) and hierarchical prefetching from the cloud to macro base stations (MBSs), enabling real-time adaptation to spatiotemporal variations in content demand across different time scales. Our goal is to minimize content transmission latency while satisfying system-wide resource and cost constraints. The proposed Vehicular-based Online Proactive caching and Prefetching (VOPP), integrates trajectory-based user mobility prediction with future content demand estimation to guide online distributed caching. At the RSU level, we formulate a distributed online convex optimization model with fine-grained gradient updates and inter-agent coordination based on real-time mobility patterns. At the MBS level, we construct a multi-step predicted content set using user mobility and request forecasts, and define a value density metric that combines popularity and delay reduction. On both levels, additional subsequent refinement steps ensure high-quality caching decisions. Extensive simulations based on a real-world GPS dataset of 10,357 taxi trajectories in Beijing demonstrate that VOPP significantly reduces transmission delay and achieves robust performance across diverse mobility patterns and user densities, outperforming baseline methods.**

***Index Terms*—proactive caching, content prefetching, online caching, vehicular edge network**

I. Introduction

The Internet of Vehicles (IoV) is gradually changing our lifestyle, enabling vehicle occupants to easily access various forms of online content. In this context, content caching is a critical concern in order to maintain high Quality of Service (QoS) levels in IoV content deliveries. By distributing content at locations near the vehicles, e.g., at macro base stations (MBS) and roadside units (RSU) provided by the Internet service providers (ISP), quick responses to user requests can be achieved. Suitable caching strategies are required to ensure that the requested content is available with low delay at the time and place where relevant requests occur.

Suitable caching decisions depend on various environmental factors such as the vehicles' geographical distribution, the users' request pattern, MBS and RSU storage and network constraints, and the contents' storage sizes. Caching strategies must jointly consider these factors while respecting overall cost constraints. This becomes particularly challenging in large-scale dynamic IoV environments, where the large numbers of MBSs and RSUs create a vast decision space and where vehicle movements can be difficult to predict.

In this paper, we tackle the challenges of distributed online proactive caching and prefetching in vehicular edge networks. Specifically, we address the problem of minimizing content transmission latency for connected vehicles under resource and cost constraints. By accurately predicting vehicle mobility and resulting content demands, we aim to proactively determine optimal caching decisions that significantly reduce user-perceived delays while adhering to limited caching resources and operational costs. Our main contributions are as follows:

- We formulate the distributed online proactive caching and prefetching problem as a hierarchical joint optimization framework, explicitly considering both content prefetching from cloud to MBSs and proactive caching decisions from MBSs to RSUs. The proposed model aims to minimize content retrieval latency while respecting capacity constraints and dynamic user mobility.
- We propose a two-level online decision-making strategy for vehicular edge networks, reformulating the model by introducing structured decision variables to capture the hierarchical interactions between MBSs and RSUs.
- At the RSU level, we formulate a distributed online convex optimization model with fine-grained gradient updates and inter-agent coordination based on real-time mobility patterns. Refinements via greedy elimination and local dynamic programming ensure constraint satisfaction and deployment robustness. At the MBS level, we construct multi-step predicted content sets and define a value density metric that combines popularity and delay reduction. A greedy-based prefetching algorithm generates efficient prefetching decisions, with a robustness-enhancing supplement to address prediction uncertainties.
- We conduct extensive performance evaluations through simulations based on real-world vehicular mobility data extracted from the Microsoft GPS dataset containing trajectories of 10,357 taxis in Beijing. Experimental results show the effectiveness and robustness of our proposed caching strategies, demonstrating significantly improved latency and caching efficiency over baseline methods.

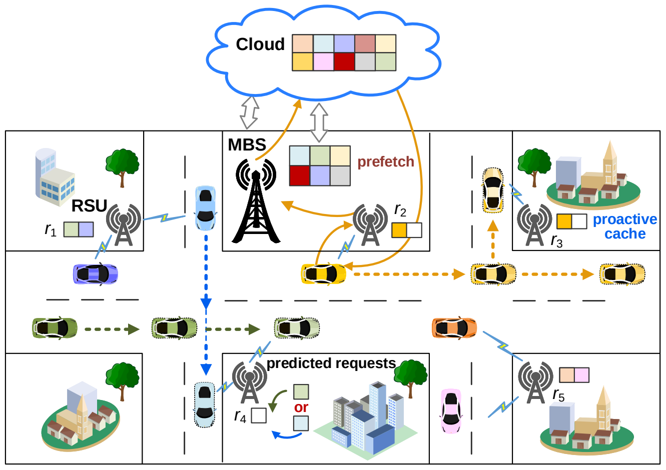


Fig. 1. An illustration of proactive and prefetching-based online caching.

II. Related work

Content caching in the IoV environment has recently attracted considerable research interest due to its significant enhancement of user QoS. The existing researches mainly focus on the reactive caching mechanism [1], [3]–[6], [15]–[17], which only caches contents when the user requests it. Zhou et al. [1] introduce a reinforcement learning framework to minimize content access cost. However, this approach does not incorporate proactive caching strategies or leverage trajectory-based user mobility prediction. Yu et al. [2] utilized a vehicle spatial and temporal trajectory prediction to realize a proactive content caching scheme. However, this method separated the environment area into a grid and updated the caching contents using a replacement strategy. Peng et al. [3] balanced the latency, traffic and workload by simulating the edge node function as a magnet that connects each other to form a larger caching pool. Liu et al. [4] distinguished peak and off-peak hours to perform an average delay minimization problem. Li et al. [5] regard the problem as a MINLP problem using the deep reinforcement learning method. Wang and Zhang et al. [6] proposed an DDQN-based algorithm to solve a joint caching and offloading problem. Wang et al. [17] proposed a popularity-incentive caching scheme to increase request availability and reduce congestion. These works perform well in a Mobile Edge Computing (MEC) content caching environment, but they can be further improved by predicting the user trace and proactively caching the requested contents.

Such a distributed content caching system can accommodate learning-based methods well [7]–[14], [18]. Feng et al. [7] proposed a federated-learning-based caching scheme with trajectory and content popularity prediction. Wu et al. [8] utilized a dueling deep Q-network to minimize the content transmission delay. Xie et al. [9] utilized a federated fine-tuning and co-inference mechanism to minimize delay and energy consumption. Liang et al. [10] proposed a federated learning framework that optimizes global and local models. Qiao et al. [11] considered client selection to minimize the objectives by proposing a distributed federal learning method. Tang et al. [12] proposed a two-timescale optimization framework considering service placing, resource provisioning and workload scheduling decisions. Li et al. [13] focused on the file priority and least recently used info as cache replacement strategies. Hu et al. [18] proposed a popularity prediction-based optimization method for cooperative caching, favoring smaller base stations and those with higher demand. However, learning-based methods demand frequent communication and model training and may thus not be applicable in some resource-constrained environments. In this work, we proposed a novel IoV content caching system by utilizing RSU proactive caching and MBS content prefetch mechanism to further improve content delivery and minimize the average transmission delay.

III. Model and Problem Formulation

A. System Model

As shown in the Figure 1, the MEC network is modeled as a three-tier caching architecture comprising the cloud layer, the edge layer, and the vehicle layer. The edge layer consists of macro base stations (MBSs) and roadside units (RSUs). We assume that the contents requested by users are initially located in the cloud data center. The service area of the RSU is covered by the MBS. We use c to represent the cloud. Let $M = \{m_j\}$ denote the set of MBSs supported by operators. Each MBS m_j has a storage capacity denoted by $\phi(m_j)$. Similarly, let $R = \{r_k\}$ be the set of RSUs, where each RSU r_k has a storage capacity denoted by $\phi(r_k)$. Let $U = \{u_i\}$ denote the user set of mobile vehicles. To capture user mobility, we assume that the system operates in a time-slotted structure, with the timeline being discretized into time slots $t \in \{1, 2, \ldots, T\}$. In this work, we assume that users move irregularly and frequently between multiple edge servers. At each time slot, vehicular nodes dynamically interact with RSUs and MBSs to complete their content requests. Additionally, we use $H = \{h_a\}$ to denote the set of variable contents provided by the operators, and the data size of each content is $\phi(h_a)$. The requests from mobile users are characterized using a mapping function $\pi : u_i \to \{1, 2, \ldots, |H|\}$, where $\pi(u_i) = a$ specifies that the content h_a is requested by the user u_i. Meanwhile, H_{r_k} and H_{m_j} denote the sets of content cached in the RSU r_k and MBS m_j.

B. Delay Model

To ensure general applicability, we assume that users' content requests remain consistent during a single macro time slot. In the system, the transmission rate between different entities is determined by the signal-to-noise ratio (SNR). The SNR between the RSU r_k and the user u_i, denoted as SNR_{r_k,u_i}, is given by $\text{SNR}_{r_k,u_i} = p_{r_k,u_i} G_{r_k,u_i} / \xi_{r_k,u_i} d^2_{r_k,u_i} \sigma^2_{r_k,u_i}$, where p_{r_k,u_i} is the transmission power, G_{r_k,u_i} is the channel gain, ξ_{r_k,u_i} is the path loss factor, d_{r_k,u_i} is the distance between the RSU and the user, and $\sigma^2_{r_k,u_i}$ is the noise power. Similarly, the SNR for the MBS m_j to the user u_i and the cloud to the MBS are also calculated based on that equation, which are defined as SNR_{m_j,u_i} and $\text{SNR}_{m_j,c}$. Furthermore, the transmission rate v_{r_k,u_i} between the RSU and the user is derived from the SNR and the allocated bandwidth b_{r_k,u_i} as $v_{r_k\to u_i} = b_{r_k,u_i}\log_2(1+\text{SNR}_{r_k,u_i})$. Similarly, the transmission rates for the MBS to user and cloud to MBS paths are $v_{m_j\to u_i}$ and $v_{c\to m_j}$, which can be obtained by using the same equation, and replacing the allocated bandwidth corresponding the b_{m_j,u_i} and $b_{m_j,c}$. The transmission delay for user u_i at time t depends on where the requested content is cached and the corresponding transmission path. We consider user content retrieval scenarios according to the location from which the requested data is delivered. First, if the requested content is cached at the RSU r_k, the user retrieves it directly from the RSU, resulting in a delay determined by the transmission rate v_{r_k,u_i}. The delay in this case is the following:

$$d^r_{u_i}(t) = x_{\pi(u_i),k}(t)\cdot\frac{k_{u_i}}{v_{r_k\to u_i}}, \tag{1}$$

where $x_{\pi(u_i),k}(t)$ is a binary variable that indicates whether the RSU r_k has cached the requested content h_a, i.e., $\pi(u_i) = a$. We use k_{u_i} to denote the size of the requested content of h_a by user u_i, where $k_{u_i} = \alpha_{u_i}\cdot\phi(h_a)$. Here, the parameter $\alpha_{u_i}\in[0,1]$ indicates that a connected vehicle user may request only a fraction of the entire content, such as a specific segment in video streaming applications, rather than the complete file.

Specifically, if the requested content is not cached in the corresponding RSU, it is forwarded to the associated MBS covering that RSU, ensuring seamless and uninterrupted delivery of data services to users of connected vehicles. If the MBS has cached the requested content, it directly transmits the content to the user u_i. In this scenario, the transmission delay is determined by the transmission rate v_{m_j,u_i} between the MBS and the user, and can be expressed as:

$$d^m_{u_i}(t) = \big(1 - x_{\pi(u_i),k}(t)\big)\cdot y_{\pi(u_i),j}(t)\cdot\frac{k_{u_i}}{v_{m_j\to u_i}}, \tag{2}$$

where $y_{\pi(u_i),j}(t)$ is a binary variable that indicates whether the MBS m_j has cached the requested content $\pi(u_i)$ of u_i.

Finally, if neither the RSU nor the MBS has cached the requested content, the MBS must retrieve the content from the cloud and then transmit it directly to the user. In this case, the total delay includes both the cloud-to-MBS transmission delay and the MBS-to-user transmission delay:

$$d^c_{u_i}(t)=\big(1-x_{\pi(u_i),k}(t)\big)\cdot\big(1-y_{\pi(u_i),j}(t)\big)\cdot\left(\frac{k_{u_i}}{v_{m_j,c}}+\frac{k_{u_i}}{v_{m_j,u_i}}\right). \tag{3}$$

The total transmission delay for the user u_i at time slot t, denoted as $D_{u_i}(t)$, is the sum of the delays from these three cases:

$$D_{u_i}(t) = d^r_{u_i}(t) + d^m_{u_i}(t) + d^c_{u_i}(t). \tag{4}$$

C. Cost Model

In addition to the transmission delay, the caching costs associated with storing and retrieving content also play a decisive role in the performance of the system. For each content h_a, the caching cost on the RSUs is denoted as $c_r(h_a) = \gamma_r\cdot\phi(h_a)$, which is linear with the size of the cached content h_a, and γ_r is the unit cost of data storage. Then, the storage cost of each RSU r_k at time slot t is given by:

$$c_{r_k}(t) = \textstyle\sum_{h_a\in H} x_{\pi(u_i),k}(t)\cdot c_r(h_a). \tag{5}$$

Similarly, for the MBS, the caching cost of h_a is defined as $c_m(h_a) = \gamma_m\cdot\phi(h_a)$, where γ_m represents the unit cost of data storage on the MBS. Thus, the storage cost of each MBS m_j at time slot t can be expressed as:

$$c_{m_j}(t) = \textstyle\sum_{h_a\in H} y_{\pi(u_i),j}(t)\cdot c_m(h_a). \tag{6}$$

The overall cost $c(t)$ for the system at time slot t as follows:

$$c(t) = \textstyle\sum_{r_k\in R} c_{r_k}(t) + \sum_{m_j\in M} c_{m_j}(t). \tag{7}$$

D. Problem Formulation

In this paper, we formulate the online caching optimization problem under the multi-layer caching architecture as a constrained optimization problem. The objective is to minimize the average transmission delay for all users over a given time period while considering the caching cost and resource constraints of the system. The calculation of the delay depends on the location of the cached content and the transmission path. The optimization problem can be expressed as follows:

$$\mathbf{P}_1 : \text{minimize}\quad \mathbf{D} = \lim_{T\to\infty}\frac{1}{T}\textstyle\sum_{t\in T}\sum_{u_i\in U} D_{u_i}(t) \tag{8}$$

$$\text{s.t.} \textstyle\sum_{h_a\in H_{r_k}}\phi(h_a)\le\phi(r_k), \sum_{h_a\in H_{m_j}}\phi(h_a)\le\phi(m_j), \tag{9}$$

$$\textstyle\sum_{t=1}^{T} c(t)\le C_{\max}, \tag{10}$$

$$x_{\pi(u_i),k}(t), y_{\pi(u_i),j}(t)\in\{0,1\}. \tag{11}$$

Equation (8) is the objective function, and equations (9) to (11) are constraints. Equation (9) is the constraint for the storage capacities of RSU and MBS. The total cost constraint over the entire caching period is formulated in Equation (10), which ensures that the cumulative caching cost over the entire period does not exceed the budget $C_{\max}$. Equation (11) characterizes whether the content requested by user u_i is cached at r_k or m_j at time slot t, where $x_{\pi(u_i),k}(t)\in\{0,1\}$, and $y_{\pi(u_i),j}(t)\in\{0,1\}$ are binary indicators denoting the caching status on the RSU and MBS, respectively.

E. Reformulation

To tackle the above optimization problem, we first change the perspective from the user level to the RSU and MBS level, where the prefetching and caching decisions are made. To this end, we introduce structured decision variables that capture the hierarchical relationships between user service requests and content caching across network layers.

We begin by modeling each RSU as an individual agent and explicitly consider the corresponding caching decisions. Here, $\mathbf{x}_k(t) = [x_{1,k}(t), x_{2,k}(t),\ldots,x_{n,k}(t)]^\top$ represents the caching decision of the contents of the k-th RSU r_k at time slot t. For each RSU r_k, the local loss function at time slot t, denoted by $l_k(\mathbf{x}_k(t))$, characterizes the performance of the service in terms of the delay perceived by the user when requesting content from r_k. According to the Equations (1) to (4), the transmission delay experienced by the mobile user u_i is determined by the location of the cached content associated with the user's request. As previously defined, $\pi(u_i) = a$

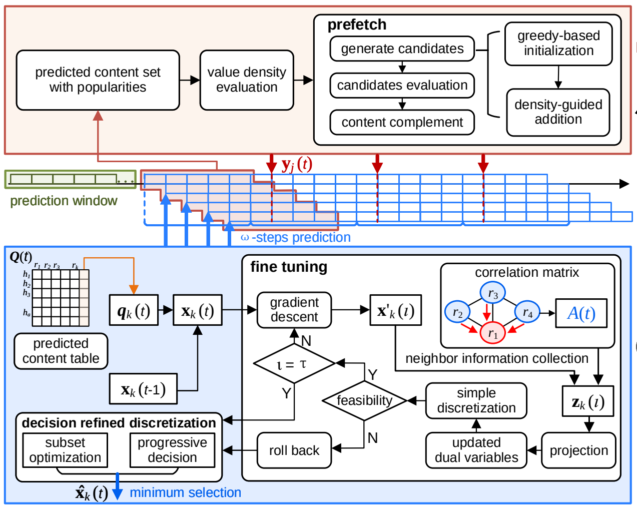


Fig. 2. The overview of the VOPP framework.

denotes that user u_i requests the content h_a, and if the request is served by RSU r_k, i.e., $x_{a,k}(t) = 1$, the transmission delay is $d^r_{u_i} = k_{u_i}/v_{r_k \to u_i}$. Otherwise, if $x_a(t) = 0$, i.e., the content is not cached at the RSU, the request is redirected to the MBS m_j that manages r_k. In this case, the transmission delay becomes either $d^m_{u_i} = k_{u_i}/v_{m_j \to u_i}$ if the content is cached at the MBS, or $k_{u_i}/v_{m_j,c} + k_{u_i}/v_{m_j \to u_i}$ if it has to be fetched from the cloud via the MBS. To facilitate the analysis and isolate the impact of RSU-side caching decisions $\mathbf{x}_k(t)$, we initially assume that all contents requested by users and associated with RSU r_k are cached in the corresponding MBS m_j, i.e., $y_{a,j}(t) = 1, \forall h_a \in H_{r_k}$. Under this assumption, the transmission delay from the MBS to the user reflects the baseline delay. We then explicitly incorporate the additional delay term $k_{u_i}/v_{m_j,c}$ in scenarios where the MBS no longer caches the requested content, i.e., when the caching decision is updated to $y_{a,j}(t) = 0$. Now, we define the local loss function to quantify the caching performance of each RSU as follows:

$$l_k(\mathbf{x}_k(t)) = \sum_{x_a \in \mathbf{x}} x_a(t) \cdot (k_{u_i}/v_{r_k \to u_i}) + (1 - x_a(t)) \cdot (k_{u_i}/v_{m_j \to u_i}), \quad (12)$$

where $x_a \in \{0, 1\}$ represents the a-th binary component of the decision vector $\mathbf{x}_k(t)$. By aggregating the local losses from all agents (i.e., RSUs) over the entire time scale, the total system loss is given by $\sum_{t=1}^{T} \sum_{k=1}^{|R|} l_k(\mathbf{x}_k(t))$. Consequently, the optimization problem on the RSU level can be reformulated as minimizing this total loss. To enforce long-term cost efficiency, we incorporate the cost constraint introduced in Equation (10). This constraint can be equivalently expressed as the following inequality constraint function:

$$g(\mathbf{x}) := \sum_{t=1}^{T} \sum_{k=1}^{|R|} c(\mathbf{x}_k(t)) - C_{\max} \leq 0. \quad (13)$$

Accordingly, the goal is to develop distributed sequential decision algorithms through which each agent selects actions that minimize the cumulative loss across RSUs. The performance of such algorithms is characterized by the regret that occurs for each agent [22], which for agent r_k is given by:

$$\mathbf{reg}(k, T) := \sum_{t=1}^{T} l_k(\mathbf{x}_k(t)) - \sum_{t=1}^{T} l_k(\mathbf{x}^*). \quad (14)$$

Algorithm 1 Module: Multi-Step User Arrival Prediction

```
Input: $U$, $R$, $\omega$;
Output: Arrival probabilities of users $\mathbf{P}(t)$;
1:  for each user $u_i \in U$ do
2:      Predict $\omega$-step future trajectory using social-LSTM;
3:      Initialize counter $\rho_i(r_k) = 0, \forall r_k \in R$;
4:      for each predicted location $\hat{s}_i(t)$ in the $\omega$-step do
5:          Find all RSUs $r_k$ such that $\hat{s}_i(t) \in$ coverage$(r_k)$;
6:          for each such $r_k$ do
7:              $\rho_i(r_k) \leftarrow \rho_i(r_k) + 1$
8:      Normalize: $p_i(r_k) \leftarrow \rho_i(r_k)/\omega$ for all $r_k \in R$;
9:      Populate the $i$-th row $\mathbf{p}_i(t)$;
10: Return $\mathbf{P}(t)$;
```

Here, $\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathbf{X}} \sum_{t=1}^{T} \sum_{k=1}^{|R|} l_k(\mathbf{x}_k(t))$. The local loss functions $l_k(\mathbf{x}_k(t))$ and the cost function $g(\mathbf{x}_k(t))$ are convex and possess bounded gradients. Specifically, we assume that there exist constants $\bar{l} > 0$ and $\bar{g} > 0$ such that: $\max_{k \in R} \max_{t \in \{1,\ldots,T\}} \max_{\mathbf{x}_k(t) \in \mathcal{X}} \|\nabla l_k(\mathbf{x}_k(t))\| \leq \bar{l}$, $\max_{t \in \{1,\ldots,T\}} \max_{\mathbf{x}_k(t) \in \mathcal{X}} \|\nabla g(\mathbf{x}_k(t))\| \leq \bar{g}$.

The regret satisfies the following bound:

***Theorem** 1:* For each agent $r_k \in R$ and $T \geq 1$, we have $\mathbf{reg}(k, T) \leq (1/2\beta_t)|R| \cdot M + \beta_t \cdot |R| \cdot \bar{l}^2 + \beta_t \cdot \bar{g}^2 C^2_{\max}(|R| - 1)^2/|R|\eta^2_{t-1}$ with constraint (10).

IV. Algorithm Design

In this section, we present the hierarchical online decision making process underlying the VOPP framework (Figure 2), which jointly addresses content caching optimization through prefetching from the cloud to the MBSs and distributed proactive caching from the MBSs to the RSUs. VOPP integrates structured decision variables to capture cross-layer dependencies and enable coordinated and latency-aware caching strategies in vehicular edge networks.

A. RSU-Agent-based Distributed Online Decision Making

To enable decentralized and timely decision making at the network edge, we reformulate the proactive caching problem into a distributed online convex optimization model. In this formulation, each RSU acts as an agent that makes caching decisions based on locally observed requests. Fine-tuning is performed based on online gradient descent on time slots and by adapting to dynamic content demand and mobility patterns while respecting local capacity constraints, ensuring scalability and responsiveness in large vehicular edge environments.

1) Mobility-aware Content Prediction: In vehicular edge networks, accurate prediction of requested content at each RSU is essential for effective proactive caching. However, uncertainty about the vehicles' mobility translates to uncertainty about future content requests. To tackle this challenge, we propose the construction of a fine-grained content demand table at the RSU level based on sequences of overlapping multi-step trajectory predictions. Specifically, we use historical vehicle trajectory data and apply the social LSTM model [21]

Algorithm 2 Mobility-aware Content Table

```
Input: $\mathbf{P}(t)$, $\pi(\cdot)$;
Output: Content-RSU probability matrix $\mathbf{Q}(t) \in \mathbb{R}^{|H|\times|R|}$;
1: for each RSU $r_k \in R$ do
2:     for each content $h_a \in H$ do
3:         $q_{a,k}(t) \leftarrow 1 - \prod_{u_i \in U, \pi(u_i)=a}(1 - p_i(r_k))$;
4: Return content-RSU probability matrix $\mathbf{Q}(t)$;
```

to predict the future positions of vehicles over a prediction window of ω time steps in a sequence of fixed-length time series. For each predicted coordinate point, we associate the user with a specific RSU if the point falls into its coverage radius.The probability of the user u_i arriving at each RSU is then obtained based on Algorithm 1. For each user $u_i \in U$, we predict a sequence of future trajectory coordinates using the social-LSTM model, and the trajectory is represented as a sequence of ω location points $\hat{s}_i(t)$. Now, we map each predicted location $\hat{s}_i(t)$ to the RSU that covers the location. A counting variable $\rho_i(r_k)$ keeps track of how often u_i is predicted to enter the coverage of r_k. The final count for each RSU is normalized by the prediction steps ω, resulting in an estimated arrival probability $p_i(r_k)$ of user u_i at RSU r_k. Finally, these probabilities are combined to form a matrix $\mathbf{P}(t)$, where each row $\mathbf{p}_i(t)$ represents the arrival distribution of the user u_i across all RSUs at time slot t.

2) RSU-level Content Demand Table Construction: To support proactive caching decisions, we construct a probabilistic requested content table $\mathbf{Q}(t)$ for each time step. For each RSU $r_k \in R$ and the content $h_a \in H$, we calculate the probability $q_{a,k}(t) \in [0, 1]$ of content h_a being requested from the RSU r_k during the time slot t, which contributes to the probability matrix between user and RSU $\mathbf{P}(t) \in \mathbb{R}^{|U|\times|R|}$. Specifically, for a given content h_a, we identify all users requesting it and aggregate their probabilities to derive the overall probability of line 3 in Algorithm 2. Then, we construct the probability vector $\mathbf{q}_k(t)$ for each RSU r_k. Finally, we obtain the complete content-RSU probability matrix $\mathbf{Q}(t)$.

3) Online Distributed Proactive Caching: Based on the predicted content RSU probability matrix $\mathbf{Q}(t)$, we introduce the distributed online caching algorithm shown in Alg. 3, which allows each RSU to proactively update its caching strategy in response to the user mobility. The algorithm operates over a sequence of macro time slots in which each RSU r_k incrementally updates its continuous caching decision vector $\mathbf{x}_k(t)$ based on historical decisions and predicted content demands. At the start of each macro time slot t, an initial decision is generated using an exponential smoothing strategy, by adding the previous decision $\mathbf{x}_k(t-1)$ with the predicted content demand probabilities $\mathbf{q}_k(t)$ by the weight value α, where $\mathbf{x}_k(t) = \alpha\mathbf{x}_k(t-1) + (1-\alpha)\mathbf{q}_k(t)$. Then, each time slot is divided into τ time slices to allow local decision refinement. Within each slice $\imath$, each RSU evaluates the local loss function and updates its decision $\mathbf{x}'_k(\imath)$ using a sub-gradient method. The update incorporates both the local gradient $\nabla l_k(\cdot)$ and a penalty term proportional to the constraint violation

Algorithm 3 Online Distributed Caching algorithm, (ODC)

```
Require: $\mathbf{Q}(t)$, $\alpha$, $\tau$, $A(t)$;
Ensure: Proactive caching strategy $\mathcal{X}$;
1: Initialize $\mathbf{x}_k(0) = \mathbf{0}$;
2: for $t = 1$ to $T$ do
3:     Initial decision $\mathbf{x}_k(t) = \alpha\mathbf{x}_k(t-1) + (1-\alpha)\mathbf{q}_k(t)$;
4:     Split time slot $t$ into $\tau$ slices;    ▷ fine-grained slices
5:     for $\imath = 0$ to $\tau$ do    ▷ fine-tuning steps
6:         Update equations (12) and (15);
7:         Each $r_k$ communicates $\mathbf{x}_k(\imath)$ to its neighbors $r_j$;
8:         Update $\mathbf{z}_k(\imath) = \sum_{j=1}^{|R|}[A(t)]_{kj}\mathbf{x}_j(\imath)$;
9:         Projection $\mathbf{x}_k(\imath+1) = \Pi_B(\mathbf{z}_k(\imath))$;
10:        Update $\lambda_k(\imath) = \frac{[g(\mathbf{x}_i(\imath))]_+}{\eta_t}$;
11:        Discretization $\hat{\mathbf{x}}_k(\imath) \leftarrow \mathbf{1}(x_a \geq \vartheta | \forall x_a \in \mathbf{x}_k)$;
12:        if constraint (9) does not hold then
13:            Break Roll back and stop tuning;
14:    Decision refining with Algorithm 4;
15: Return $\mathcal{X} = \{\hat{\mathbf{x}}_k(t) | t \in \{1, \ldots, T\}\}$;
```

Algorithm 4 decision refining

```
Require: decision $\mathbf{x}_k(t)$;
Ensure: decision $\hat{\mathbf{x}}_k(t)$;
1: $\hat{\mathbf{x}}_k \leftarrow \mathbf{1}(x_a \neq 0 | \forall x_a \in \mathbf{x}_k)$;
2: if $\mathbf{x}_k = \mathbf{0}$ then
3:     Return $\hat{\mathbf{x}}_k = \mathbf{0}$
4: while constraint (9) not holds do
5:     $\arg\min_{x_a \in \mathbf{x}_k(t)}\{\mathbf{x}_k(t)\} = 0$;
6: Calculate $\mathbf{x}^*$ based on DP using prediction values;
7: $\hat{\mathbf{x}}_k(t) \leftarrow \mathbf{x}_k(t) | \arg\min_{\mathbf{x}\in\{\mathbf{x}_k(t),\mathbf{x}^*\}} l_k(\mathbf{x})$;
8: Return $\hat{\mathbf{x}}_k$.
```

$[g(\mathbf{x}_k(\imath))]_+$, modulated by a dual variable $\lambda_k(\imath)$, where

$$\mathbf{x}'_k(\imath) = \mathbf{x}_k(\imath) - \beta_\imath[\nabla l_k(\mathbf{x}_k(\imath)) + \lambda_k(\imath)\partial[g(\mathbf{x}_k(\imath))]_+]. \quad (15)$$

To enable distributed coordination, each r_k incorporates the decisions of its neighboring RSUs to form an aggregated intermediate decision vector $\mathbf{z}_k(\imath)$. The aggregation progress is performed via a weighted averaging process governed by dynamically constructing a row-stochastic influence matrix $[A(t)]_{kj}$ based on real-time vehicular mobility patterns. Specifically, each element is determined by the number of vehicles moving from r_j to r_k. These transition frequencies are normalized to ensure that each row of $[A(t)]_{kj}$ sums to 1, thus the convexity of the update step is preserved. After that, the aggregated decision vector is then projected onto the feasible continuous region $B^{|H|}$, where $x_a \in B^{|H|}$, to ensure compliance with memory constraints, yielding the updated decision $\mathbf{x}_k(\imath+1)$ in line 9. In line 10, we update the dual variable. Afterwards, to ensure deployable binary caching strategies, the continuous vector is further discretized via thresholding to obtain $\hat{\mathbf{x}}_k(\imath)$. A feasibility check is performed by discretization to verify compliance with capacity. If the decision violates constraints, the algorithm rolls back to the last feasible state and stops further fine-tuning progress within the current large time slot. Finally, we apply a decision refinement module, as

Algorithm 5 greedy-based complementary prefetch

```
Require: $\phi(h_a)$, $\mathcal{N}(h_a)$, $\Delta(h_a)$ $\forall h_a \in H$;
Ensure: decision $\mathbf{y}_j(t)$;
1: Calculate value density $V_j(h_a)$ of each content $h_a$;
2: Construct content list $\mathbb{I}_{m_j}$ by descending order of $V_j(h_a)$;
3: for $h_a \in \mathbb{I}_{m_j}$ do          ▷ initial greedy caching decision
4:     if $\phi(h_a) \leq \phi(m_j) - \sum_{h_b \in H_{m_j}} \phi(h_b)$ then
5:         $H_{m_j} \leftarrow H_{m_j} \cup \{h_a\}$;
6: Construct $H_{m_j}^{\mathtt{top}(k)} = \{h_a | \arg\max_{h_a \in H}^k V_j(h_a)\}$;
7: $\hat{H}_{m_j} \leftarrow H_{m_j} \cup H_{m_j}^{\mathtt{top}(k)}$;
8: $H_{m_j} \leftarrow$ local optimized using $\hat{H}_{m_j}$ as the object set;
9: $\mathbf{y}_j(t) \leftarrow \mathbf{1}(y_j \in H_{m_j} | \forall y_j \in \mathbf{y}_j)$;
10: $\mathbb{L}_{m_j} = \{h_a | y_a = 0, \arg\max \Delta(h_a)\}$
11: for $h_a \in \mathbb{L}_{m_j}$ do          ▷ remaining storage supplement
12:     if $\phi(h_a) \leq \phi(m_j) - \sum_{h_b \in H_{m_j}} \phi(h_b)$ then
13:         $y_a = 1, y_a \in \mathbf{y}_j(t)$;
14: Return $\mathbf{y}_j(t)$
```

shown in Algorithm 4, to improve the quality of the discrete decision, further refining the caching efficiency. The algorithm returns a sequence of binary caching decisions in $\mathbb{B}^{|H|}$, where $\hat{\mathbf{x}}_k(t)|t \in \{1, \ldots, T\}$ across all RSUs.

To further refine the caching decision obtained from the local optimization at each macro time slot, we propose the decision discretization strategy shown in Alg. 4, which integrates a greedy elimination procedure and local dynamic programming based on the content loss contribution to ensure constraint satisfaction while minimizing the total delay. We start by discretizing the continuous decision vector from the subgradient optimization using a greedy removal strategy (lines 1 to 5). First, the content with non-zero values $x_a \in \mathbf{x}_k \neq 0$ is added, and the algorithm iteratively discards the content with the smallest decision weight $\arg\min_{x_a \in \mathbf{x}_k(t)} \{\mathbf{x}_k(t)\}$ until the caching constraint (9) is satisfied, whereby an intermediate, feasible discrete solution can be obtained. To further improve this greedy solution, we propose a local dynamic programming (DP) method based on the subset of candidate contents according to their contribution to the total loss. We introduce the notion of content loss contribution, which quantifies the benefit of caching this content on a given RSU.

***Definition** 1 (content loss contribution):* The content loss contribution, i.e., the loss reduction when caching content h_a on RSU r_k is $\Phi(h_a, r_k) = l_k(\mathbf{0}) - l_k(\mathbf{x}|_{x_a=1, x_b=0, h_b \neq h_a})$.

The subset of contents that were marked as the relaxed decision ($x_a > 0$) is selected as the candidate set, which are treated as DP objects, where the object size corresponds to the content size and the object value is derived from the estimated delay reduction based on the predicted user demands $\Phi(h_a, r_k)$. Having obtained both the greedy $\mathbf{x}_k(t)$ and the refined $\mathbf{x}^*$ generated by the DP, we evaluate the corresponding objective loss for each candidate. The candidate with the lower predicted loss is selected as the final decision $\hat{\mathbf{x}}_k(t)$.

B. MBS-level Prefetch

1) Content Popularity Prediction: To enable effective prefetching at the macro base station (MBS) level, we construct a predicted content set based on future user trajectories and their service requests. Since each user u_i produces an ω-step predicted trajectory window over future time slots, we aggregate these trajectories and associate each predicted location with its corresponding serving MBS. Based on the user-content mapping, all content requests predicted to fall within the service coverage of a given MBS m_j are aggregated to form a predicted content set H_{m_j}, which is performed over an ω-micro slot prediction window, reflecting the fact that MBSs operate on a lower update frequency compared to RSUs. Specifically, the MBS caching decisions, denoted by $\mathbf{y}_j(t)$, are updated every ω time slots rather than at each individual time slot. This prediction and aggregation process is visually highlighted in red in Figure 2. Subsequently, we quantify the popularity $\mathcal{N}(h_a)$ of each content in the predicted set by counting the number of times it is expected to be requested under the coverage of the same MBS, which reflects the aggregated demands within the prediction window.

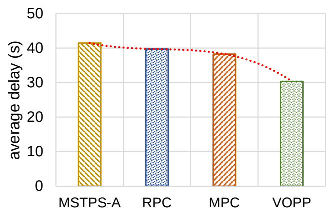
2) Value Density Evaluation: To make content prefetching decisions, we treat the selection of contents for each MBS as a 0-1 knapsack problem, where each content item is modeled as an object with a cost (storage size) and a benefit (delay reduction based on predicted requests). The objective is to maximize the overall delay reduction under the MBS storage constraint according to the prediction windows. To simplify the notation, we use $D_{u_i}(t)^{(0,0)}$ to denote the delay of u_i at time slot t when content h_a is not cached at either the RSU or the MBS, where $D_{u_i}(t)^{(0,0)} = D_{u_i}(t)|_{x_{a,k}=0, y_{a,j}=0}$. Similarly, $D_{u_i}(t)^{(0,1)}$ represents the delay when content is not cached in the RSU but is available at the MBS, where $x_{a,k} = 0$, $y_{a,j} = 1$. Then, the delay reduction is defined as

***Definition** 2 (delay reduction):* Let $\Delta(h_a)$ be the delay reduction of h_a, which quantifies the transmission delay benefit when the content is cached at the macro base station (MBS) instead of being fetched from the remote cloud, where

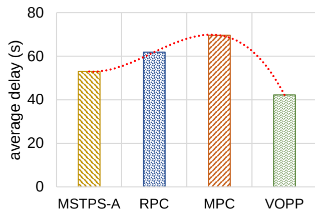
$$\Delta(h_a) = [D_{u_i}(t)^{(0,0)} - D_{u_i}(t)^{(0,1)}]|_{u_i \in U | \pi(u_i) = a}. \quad (16)$$

***Definition** 3 (value density):* The content value density $V_j(h_a)$, which quantifies the caching worthiness of the content h_a on a given MBS m_j, is: $V_j(h_a) = \mathcal{N}(h_a) \cdot \Delta(h_a) / \phi(h_a)$, where $s(h_a)$ is the content size, and $\mathcal{N}(h_a)$ represents the content popularity.

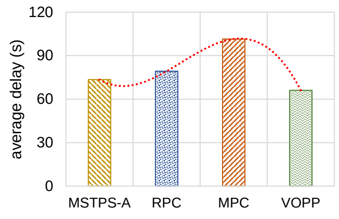
The MBS caching decision is generated by a hybrid strategy combining greedy selection with local dynamic programming correction described in Algorithm 5. We start with an initial greedy phase in lines 2 to 5, where the contents are selected in descending order of their value density $V_j(h_a)$ until the storage capacity of the MBS m_j is reached. On this basis, we construct a candidate set $\hat{H}_{m_j}^{\mathtt{top}(k)}$ by combining the initial greedy selection with the top k contents with the highest value density, where $\hat{H}_{m_j}^{\mathtt{top}(k)} = \arg\max_{h_a \in H}^k V_j(h_a)$. Then, we identify the optimal subset from $\hat{H}_{m_j}^{\mathtt{top}(k)}$ that maximizes the cumulative value under the capacity constraint. Here, the value density



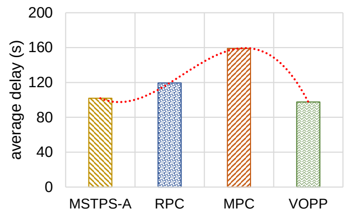
(a) # of users 100.



(b) # of users 150.

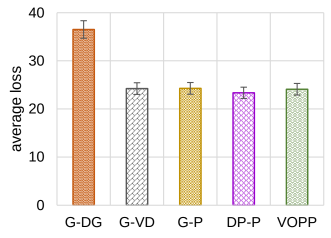


(c) # of users 200.

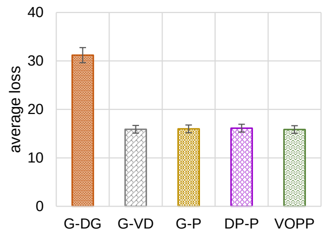


(d) # of users 264.

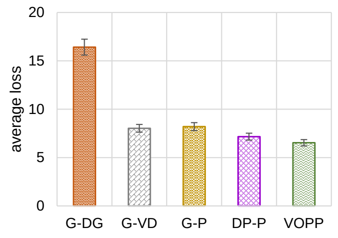
Fig. 3. obj. (average latency) for different numbers of users.



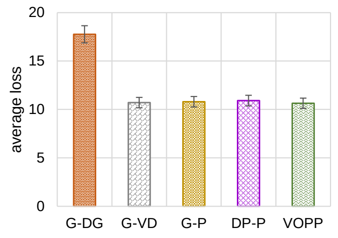
(a) $\phi(m_j) = 5$G.



(b) $\phi(m_j) = 6$G.



(c) $\phi(m_j) = 7$G.



(d) $\phi(m_j) = 8$G.

Fig. 4. obj. (average latency) for different sizes of MBSs.

is treated as object utility and the content size as weight, which enables a refined selection beyond greedy heuristics. However, due to potential deviation between the predicted and real user distribution, relying only on the locally optimized DP decision may result in suboptimal cache performance in practice. Specifically, under the predicted demand, content requested by unexpectedly arriving users may be missed. Therefore, we carry out a supplementary search step that aims to improve the robustness of the caching strategy by filling the unused memory based on the evaluation of the content delay gains $\Delta(h_a)$ (lines 8 to 10). Subsequently, we obtain the final decision $\mathbf{y}_j(t)$ from the caching set H_{m_j} for the MBS.

V. Evaluations

A. Basic Setting

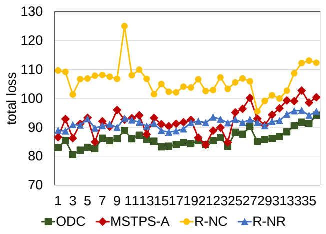
We conduct our experiments using a real-world mobility dataset provided by Microsoft, which records GPS traces from 10,357 taxis operating in Beijing [19], [20]. To focus on a dense urban area, we mark out a 2.9 km^2 square area centered around Financial Street, Beijing as the center point (coordinates [116.36032115, 39.911045075]). The selected area spans a longitude range from 116.3518143 to 116.368828 and a latitude range from 39.8996048 to 39.92248535. First, we implemented coarse-grained filtering of the trajectory points through the entire dataset inside this area, and we got 8432 users. To ensure the temporal continuity and richness of each trajectory, we apply a fine-grained filtering criterion by discarding any trajectory with fewer than 50 GPS points. In addition, the simulation environment is configured with multiple RSUs and MBSs deployed within the selected area. The RSUs are evenly distributed and configured with constrained storage and computing capacity to approximate realistic vehicular edge network conditions. The MBSs are strategically placed to cover multiple RSUs and operate at a lower update frequency.

B. Experiment Results

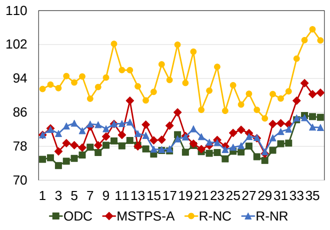
1) Evaluation of Average Delay: We compare the proposed VOPP framework against three baselines: MSTPS-based Adaptation (MSTPS-A) [2], RSU Proactive Caching (RPC), and MBS Prefetch Caching (MPC) for different user counts from 100 to 264, and show consistent effectiveness in reducing the overall delay in different scenarios. The results in Figure 3 show that the VOPP framework demonstrates adaptivity to sparse user distributions and effectively maintains caching performance even with a limited number of users. For example, at a user scale of 100, VOPP achieves an average delay of approximately 30.34 seconds, significantly outperforming MSTPS-A and RPC, both of which require around 40 seconds. This performance gap highlights the robustness of VOPP in scenarios where proactive caching and prefetching could be expected to degrade due to insufficient users. Moreover, as the number of users increases, the delay of VOPP remains lower compared to all baselines.

We conducted extensive experiments with varying numbers of users ranging from 100 to 264 users to systematically evaluate the performance of VOPP against several representative baselines. Here, the MSTPS-A method performs well in high user number scenarios but struggles to perform well in a low user number scenario. We ascribe this observation to the low user number causing a low number of in-group users for each RSU, resulting in inaccurate caching. The MPC method becomes overwhelmed when facing a large number of users. Because the MPC method considers the MBS layer prefetching strategies without conducting proactive caching in the RSU layer, the method cannot take full advantage of the RSU resources, increasing the burden on the MBS layer. Thus, the MPC method gets worse when the number of users increases. The results show that VOPP consistently has a lower average transmission delay for different numbers of users and outperforms other algorithms. As the number of users increases, MSTPS-A improves in performance but is still slightly inferior to VOPP.

2) MBS-layer Evaluation: We compare the performance of four representative caching strategies implemented at the MBS layer to evaluate the effectiveness of our proposed



(a) normal (1), large (3).



(b) normal (2), large (4).

Fig. 5. obj. (average latency) for different sizes of RSUs.

content prefetching method, and the baselines include Greedy based on Value Density (G-VD), DP based on multiple-step Prediction trajectories (DP-P), Greedy based on Popularity (G-P), and Greedy based on Delay Gain (G-DG). We evaluate the performance of different MBS-level caching strategies under varying memory capacities. As shown in Figure 4, the G-DG baseline consistently performs significantly worse than the other methods across all memory configurations. This indicates that delay gain alone is not sufficient to effectively guide caching decisions. In contrast, strategies such as G-VD and DP-P, which consider both popularity and delay properties, show better adaptability in content selection. We then performed extensive experiments with MBS memory size, where the average transmission delay decreases for all strategies. This is expected since a larger cache can store more high-quality content, reducing the likelihood of costly content retrieval from remote sources.

3) RSU-layer Evaluation: We evaluate the RSU-layer performance by comparing the proposed VOPP method's ODC algorithm (cf. Algorithm 3) with three baselines: MSTPS-based Adaptation (MSTPS-A) [2], RSU without Neighbor Cooperation (R-NC), and RSU without Refinement (R-NR) under different RSU memory size. RSUs with a large average number of users were treated as "large" RSU, while the others were treated as "normal" RSUs, with a "normal 2G / large 4G" setting describing a sufficient RSU-layer capacity and a "normal 1G / large 3G" setting describing a capacity shortage. From the results in Figure 5, the VOPP method shows a consistent and superior performance both on sufficient and insufficient RSU-layer capacity situations. The MSTPS-A method reveals the effectiveness of the probability-based caching idea but is still slightly worse than the proposed VOPP method. The R-NC method disabled the communication between RSUs so that RSUs cannot proactively foresee the incoming users from their neighbor RSUs. This results in an isolated caching that only depends on the LSTM's predictions and performs badly in the evaluation. The R-NR method replaced the decision refining by a simple discretization and feasibility check method, which is blind to the content priorities and fails to trade off those contents under the constraint of memory size, resulting in poor performance.

VI. Conclusion

This paper addresses the challenge of minimizing transmission delay in vehicular edge networks through a hierarchical framework for distributed online proactive caching and prefetching. We propose VOPP, a two-level optimization framework that jointly considers proactive caching at RSUs and prefetching from the cloud to MBSs, guided by multi-step trajectory-based demand prediction. The RSU-level decision-making adopts a distributed online convex optimization model with coordination among agents and a decision enhancement mechanism combining greedy elimination and dynamic programming. At the MBS level, we construct a predicted content set using user mobility forecasts and employ a hybrid greedy-complementary algorithm to generate robust prefetching decisions under uncertainty. Extensive experiments on a real-world GPS dataset demonstrate that VOPP significantly reduces transmission delay and consistently outperforms baseline methods across varying user densities and mobility patterns. Future work will explore joint learning-based optimization for content popularity and mobility pattern estimation, and further extend the framework to handle federated scenarios with multi-domain coordination and privacy constraints.

References

[1] Zhou, H., Jiang, K., He, S., Min, G., & Wu, J. (2023). Distributed deep multi-agent reinforcement learning for cooperative edge caching in internet-of-vehicles. IEEE Transactions on Wireless Communications, 22(12).

[2] Yu, G., He, Y., Wu, J., Chen, Z., & Pan, J. (2023). Mobility-aware proactive edge caching for large files in the internet of vehicles. IEEE Internet of Things Journal, 10(13).

[3] Peng, J., Li, Q., Tang, X., Zhao, D., Hu, C., & Jiang, Y. (2023). A cooperative caching system in heterogeneous edge networks. IEEE Transactions on Mobile Computing, 23(7).

[4] Liu, L., Yuan, X., Zhang, N., Chen, D., Yu, K., & Taherkordi, A. (2023). Joint computation offloading and data caching in multi-access edge computing enabled internet of vehicles. IEEE Transactions on Vehicular Technology, 72(11).

[5] Li, Z., Yang, C., Huang, X., Zeng, W., & Xie, S. (2023). CoOR: Collaborative task offloading and service caching replacement for vehicular edge computing networks. IEEE Transactions on Vehicular Technology, 72(7).

[6] Wang, L., & Zhang, G. (2023). Joint service caching, resource allocation and computation offloading in three-tier cooperative mobile edge computing system. IEEE Transactions on Network Science and Engineering, 10(6).

[7] Feng, B., Feng, C., Feng, D., Wu, Y., & Xia, X. G. (2023). Proactive content caching scheme in urban vehicular networks. IEEE Transactions on Communications, 71(7).

[8] Wu, Q., Zhao, Y., Fan, Q., Fan, P., Wang, J., & Zhang, C. (2022). Mobility-aware cooperative caching in vehicular edge computing based on asynchronous federated and deep reinforcement learning. IEEE Journal of Selected Topics in Signal Processing, 17(1).

[9] Xie, G., Xiong, Z., Zhang, X., Xie, R., Guo, S., Guizani, M., & Poor, H. V. (2024). Gai-iov: Bridging generative ai and vehicular networks for ubiquitous edge intelligence. IEEE Transactions on Wireless Communications, 23(10).

[10] Liang, J., Li, J., Zhang, J., & Zang, T. (2024, June). Federated Learning with Data-Free Distillation for Heterogeneity-Aware Autonomous Driving. In 2024 IEEE International Joint Conference on Neural Networks (IJCNN).

[11] Qiao, D., Guo, S., Liu, D., Long, S., Zhou, P., & Li, Z. (2022). Adaptive federated deep reinforcement learning for proactive content caching in edge computing. IEEE Transactions on Parallel and Distributed Systems, 33(12).

[12] Tang, L., Xu, M., Xu, C., & Ye, K. (2024, July). Collaborative Resource Management and Workloads Scheduling in Cloud-Assisted Mobile Edge Computing across Timescales. In 2024 IEEE International Conference on Web Services (ICWS).

[13] Li, C., Song, M., Du, S., Wang, X., Zhang, M., & Luo, Y. (2020). Adaptive priority-based cache replacement and prediction-based cache prefetching in edge computing environment. Journal of Network and Computer Applications, 165.

[14] Qiao, G., Leng, S., Maharjan, S., Zhang, Y., & Ansari, N. (2019). Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks. IEEE Internet of Things Journal, 7(1).

[15] Chen, X., Chen, S., Ni, W., Wang, X., Zhang, S., Zhang, & Jamalipour, A. (2024). Optimal two-timescale configuration of mobile edge computing with mixed energy supply. IEEE Transactions on Smart Grid, 15(5).

[16] Wang, C., Chen, C., Pei, Q., Lv, N., & Song, H. (2020). Popularity incentive caching for vehicular named data networking. IEEE Transactions on Intelligent Transportation Systems, 23(4).

[17] Wang, C., Chen, C., Pei, Q., Lv, N., & Song, H. (2020). Popularity incentive caching for vehicular named data networking. IEEE Transactions on Intelligent Transportation Systems, 23(4).

[18] Hu, Z., Fang, C., Wang, Z., Tseng, S. M., & Dong, M. (2023). Many-objective optimization based-content popularity prediction for cache-assisted cloud-edge-end collaborative IoT networks. IEEE Internet of Things Journal, 11(1).

[19] Yuan, J., Zheng, Y., Xie, X., & Sun, G. (2011, August). Driving with knowledge from the physical world. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[20] Yuan, J., Zheng, Y., Zhang, C., Xie, W., Xie, X., Sun, G., & Huang, Y. (2010, November). T-drive: driving directions based on taxi trajectories. In Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems.

[21] Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., & Savarese, S. (2016). Social lstm: Human trajectory prediction in crowded spaces. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[22] Yuan, D., Proutiere, A., & Shi, G. (2021). Distributed online optimization with long-term constraints. IEEE Transactions on Automatic Control, 67(3).

VII. Appendix

A. *Proof of Theorem 1*

Proof: We start with the evolution of $||\mathbf{x}_k(t+1)-\mathbf{x}^*||^2$.

$$||\mathbf{x}_k(t+1)-\mathbf{x}^*||^2 = ||\Pi_B(\mathbf{z}_k(t))-\mathbf{x}^*||^2 \leq ||\mathbf{z}_k(t)-\mathbf{x}^*||^2 \quad (17)$$

Since $\mathbf{z}_k(t) = \sum_{j=1}^{|R|}[A(t)]_{kj}\mathbf{x}_j(t)$, and for each time slot, $\mathbf{x}'_k(t) = \mathbf{x}_k(t) - \beta_t[\nabla l_k(\mathbf{x}_k(t)) + \lambda_k(t)\partial[g(\mathbf{x}_k(t))]_+]$. In order to design and bound the analysis, we perform the reconstruction by introducing the online augmented Lagrangian function:

$$L_k(\mathbf{x}_k(t),\boldsymbol{\lambda}_k(t)) \overset{\triangle}{=} l_k(\mathbf{x}_k(t))+\boldsymbol{\lambda}_k(t)[g(\mathbf{x}_k(t))]_+-(\eta_t/2)\boldsymbol{\lambda}_k(t)^2, \quad (18)$$

where $-\eta_t/2\boldsymbol{\lambda}_k(t)^2$ is the regularization term. To simplify the presentation, we use $\triangledown_k(t)$ to denote the gradient of the augmented function where $\triangledown_k(t) = \triangledown_{\mathbf{x}}L_k(\mathbf{x}_k(t),\boldsymbol{\lambda}_k(t))$. Then, we have

$$\begin{aligned}
&||\mathbf{x}_k(t+1)-\mathbf{x}^*||^2 \leq ||\textstyle\sum_{j=1}^{|R|}[A(t)]_{kj}[x_j(t)-\beta_t\triangledown_k(t)]-\mathbf{x}^*||^2\\
&\leq \textstyle\sum_{j=1}^{|R|}||[A(t)]_{kj}[x_j(t)-\beta_t\triangledown_k(t)-\mathbf{x}^*]||^2\\
&\leq \textstyle\sum_{j=1}^{|R|}||[A(t)]_{kj}||^2\sum_{j=1}^{|R|}||[x_j(t)-\beta_t\triangledown_k(t)-\mathbf{x}^*]||^2\\
&\leq \textstyle\sum_{j=1}^{|R|}||[x_j(t)-\beta_t\triangledown_j(t)-\mathbf{x}^*]||^2 \qquad (19)
\end{aligned}$$

$[A(t)]_{kj}$ is a row stochastic matrix, i.e., the sum of each row is 1. Then, we have $||\sum_{j=1}^{|R|}[A(t)]_{kj}[x_j(t)-\beta_t\triangledown_k(t)]-\mathbf{x}^*||^2 = ||\sum_{j=1}^{|R|}[A(t)]_{kj}[x_j(t)-\beta_t\triangledown_k(t)-\mathbf{x}^*]||^2$. According to the Cauchy inequality, we can have equation (19). Since k and j are used as indices here, the summation result does not change, so we obtain

$$||\mathbf{x}_k(t+1)-\mathbf{x}^*||^2 \leq \textstyle\sum_{k=1}^{|R|}||[\mathbf{x}_k(t)-\mathbf{x}^*-\beta_t\triangledown_k(t)]||^2 \quad (20)$$

The right side of the equation reduces further to $\sum_{k=1}^{|R|}||[\mathbf{x}_k(t)-\mathbf{x}^*-\beta_t\triangledown_k(t)]||^2 = \sum_{k=1}^{|R|}||\mathbf{x}_k(t)-\mathbf{x}^*||^2 + \beta_t^2\sum_{k=1}^{|R|}||\triangledown_k(t)||^2 - 2\beta_t\sum_{k=1}^{|R|}\triangledown_k^\top(t)(\mathbf{x}_k(t)-\mathbf{x}^*)$ From the definitions above, $\triangledown_k^\top(t)(\mathbf{x}_k(t)-\mathbf{x}^*)$ is the simplified notation of $\triangledown_{\mathbf{x}}L_k(\mathbf{x}_k(t),\boldsymbol{\lambda}_k(t))(\mathbf{x}_k(t)-\mathbf{x}^*)$, where $\triangledown_{\mathbf{x}}L_k(\mathbf{x}_k(t),\boldsymbol{\lambda}_k(t)) \approx [L_k(\mathbf{x}_k(t)+h,\boldsymbol{\lambda}_k(t)) - L_k(\mathbf{x}_k(t),\boldsymbol{\lambda}_k(t))]/h$. Due to the function's convexity, one can deduce that $\triangledown_k^\top(t)(\mathbf{x}_k(t)-\mathbf{x}^*) \leq L_k(\mathbf{x}_k(t),\boldsymbol{\lambda}_k(t)) - L_k(\mathbf{x}^*,\boldsymbol{\lambda}_k(t))$. With the online augmented Lagrangian function (18) yields

$$\begin{aligned}
&\textstyle\sum_{k=1}^{|R|}[l_k(\mathbf{x}_k(t)) + \lambda_k(t)[g(\mathbf{x}_k(t))]_+ - (\eta_t/2)\lambda_k(t)^2 -\\
&(l_k(\mathbf{x}^*) + \lambda_k(t)[g(\mathbf{x}^*)]_+ - (\eta_t/2)\lambda_k(t)^2)]\\
&= \textstyle\sum_{k=1}^{|R|}[l_k(\mathbf{x}_k(t))-l_k(\mathbf{x}^*)] + \sum_{k=1}^{|R|}\lambda_k(t)([g(\mathbf{x}_k(t))]_+ - [g(\mathbf{x}^*)]_+). \qquad (21)
\end{aligned}$$

Based on Equation (13), we obtain $[g(x^*)]_+ = 0$. In addition, since $[g(\mathbf{x}_k(t))]_+ \geq 0$, then we have

$$\begin{aligned}
&\textstyle\sum_{k=1}^{|R|}[l_k(\mathbf{x}_k(t)) - l_k(\mathbf{x}^*)] \leq (1/2\beta_t)\sum_{k=1}^{|R|}||\mathbf{x}_k(t)-\mathbf{x}^*||^2\\
&-(1/2\beta_t)||\mathbf{x}_k(t+1)-\mathbf{x}^*||^2 + (\beta_t/2)\textstyle\sum_{k=1}^{|R|}||\triangledown_k(t)||^2. \qquad (22)
\end{aligned}$$

Since the squared Euclidean norm is non-negative and with a positive scaling factor $\beta_t \geq 0$, this expression satisfies $(1/2\beta_t)||\mathbf{x}_k(t+1)-\mathbf{x}^*||^2 \geq 0$. In addition, for each RSU r_k, the caching decision $\mathbf{x}_k(t)$ must satisfy the storage capacity constraint of Equation (9) in each time slot t. We use M to denote the length of the decision vector $\mathbf{x}_k(t)$, which means the maximum number of contents that can be cached in each RSU. Consequently, the deviation between the caching decision and the optimal solution satisfies $||\mathbf{x}_k(t)-\mathbf{x}^*||^2 \leq M$, and by considering all RSUs we obtain the bound $\sum_{k=1}^{|R|}||\mathbf{x}_k(t)-\mathbf{x}^*||^2 \leq |R|\cdot M$. Therefore, we have

$$\textstyle\sum_{k=1}^{|R|}[l_k(\mathbf{x}_k(t))-l_k(\mathbf{x}^*)] \leq \frac{1}{2\beta_t}|R|\cdot M + \frac{\beta_t}{2}\sum_{k=1}^{|R|}||\triangledown_k(t)||^2. \quad (23)$$

Let $\bar{l}$ and $\bar{g}$ denote the maximum of the gradient of $l_k(\mathbf{x}_k(t)), \forall k,t$ and $g(\mathbf{x}_k(t))$. For the last term of the right side of the equation, $||\triangledown_k(t)||^2$ further reduces to

$$\begin{aligned}
||\triangledown_k(t)||^2 &= \|\triangledown_{\mathbf{x}}L_k(\mathbf{x}_k(t),\lambda_k(t))\|^2\\
&= \|\nabla l_k(\mathbf{x}_k(t)) + \lambda_k(t)\partial[g(\mathbf{x}_k(t))]_+\|^2\\
&\leq 2\|\nabla l_k(\mathbf{x}_k(t))\|^2 + 2\lambda_k(t)^2\|\partial[g(\mathbf{x}_k(t))]_+\|^2\\
&\leq 2\bar{l}^2 + 2\bar{g}^2\cdot\lambda_k(t)^2 = 2\bar{l}^2 + 2\bar{g}^2\cdot[g(\mathbf{x}_k(t))]_+^2/\eta_{t-1}^2
\end{aligned}$$

Since the value of $[g(\mathbf{x}_k(t))]_+$ characterizes the violation of the cost constraint in each time slot t, and considering the long-term cost constraint, the cumulative caching cost for all RSUs over the entire time scale $\{1,\ldots,T\}$ cannot be higher than $C_{\max}$. Thus, for each agent (RSU), at any given time slot t, we have $[g(\mathbf{x}_k(t))]_+ \leq \bar{C}_r(|R|-1)/|R|$. Then, we have

$$||\triangledown_k(t)||^2 \leq 2\bar{l}^2 + 2\bar{g}^2\cdot C_{\max}^2(|R|-1)^2/|R|^2\eta_{t-1}^2 \quad (24)$$

Therefore, combining the inequalities (23) and (24), we have

$$\begin{aligned}
&\textstyle\sum_{k=1}^{|R|}[l_k(\mathbf{x}_k(t)) - l_k(\mathbf{x}^*)] \leq (1/2\beta_t)|R|\cdot M + \beta_t\cdot|R|\cdot\bar{l}^2 +\\
&\beta_t\cdot\bar{g}^2C_{\max}^2(|R|-1)^2/|R|\eta_{t-1}^2. \qquad (25)
\end{aligned}$$

■