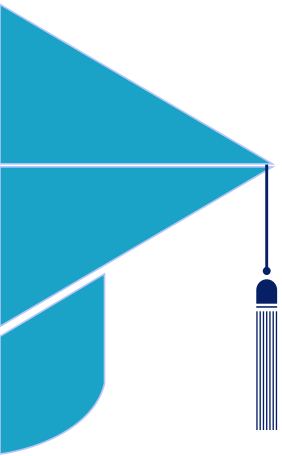
The 34th International Conference on Computer Communications and Networks (ICCCN 2025)



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

Shuaibing Lu, **Bojin Xiang**
College of Computer Science
Beijing University of Technology
Beijing, China

Jie Wu
Center for Networked Computing
Temple University
Philadelphia, USA

Philipp Andelfinger, Wentong Cai
College of Computing and Data Science
Nanyang Technological University
Singapore

Emerging IoV services

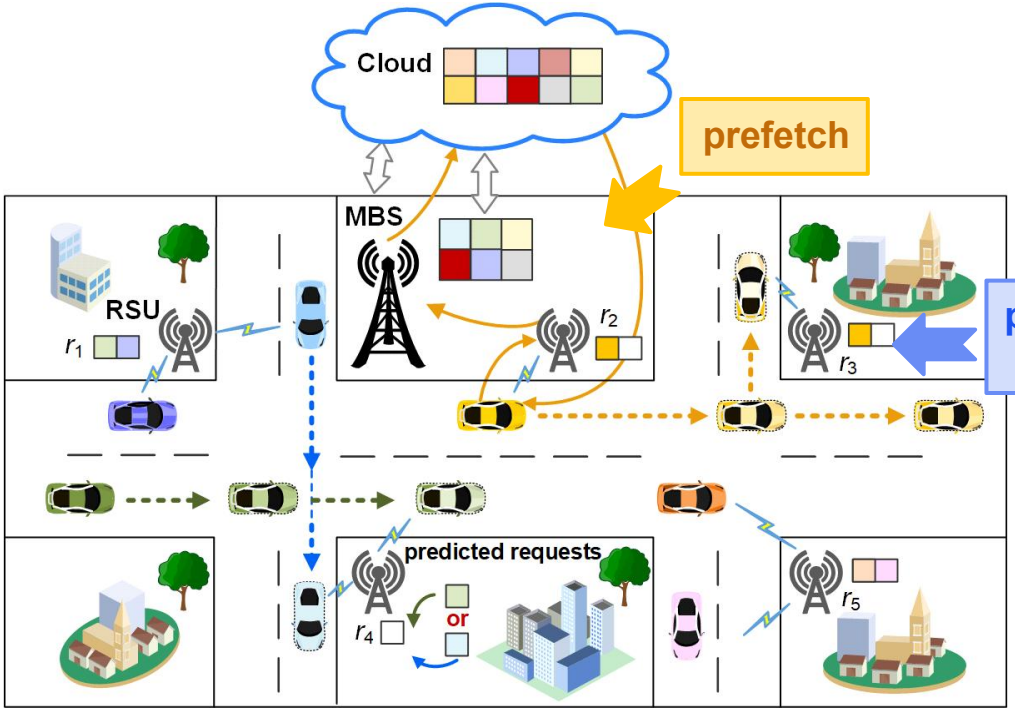
- Internet of Vehicles (IoV) technologies enable diverse time-sensitive applications.
- Low-latency services rely on proactive content distribution at MBSs and RSUs.
- User mobility and random requests complicate caching in dynamic IoV systems.

Automated Driving

Multimedia

AR navigation

An example - scenario



node	storage capability
r_1	3
r_2	6
r_3	2
r_4	1
r_5	2
MBS	20

link	$v_{user \to r_1}$	$v_{user \to r_2}$	$v_{user \to r_3}$	$v_{user \to r_4}$	$v_{user \to r_5}$	$v_{r_1 \to MBS}$
bandwidth	3	5	4	3	1	2

- resource-constrained
- vehicle mobility
- diverse requests
- different update frequencies

Challenges

- **C1. Proactive Cross-tier Content Caching**

  How to create a strategy that prefetches content from the cloud to MBS and proactively caches the content onto the RSUs?

- **C2. Mobility and Demand Uncertainty**

  How can we balance the trade-off between minimizing response time to ensure user QoS and avoiding the overload of limited resources?

Model and Formulation

System Model

entity	notation
MBS	$M = \{m_j\}$
RSU	$R = \{r_k\}$
content	$H = \{\mathrm{h}_a\}$
vehicle (user)	$U = \{u_i\}$

indication	notation
caching decision	H_{m_j}, H_{r_k}
cache indicator	$x_{\pi(u_i),k}(t), y_{\pi(u_i),j}(t)$
u_i requesting for h_a	$\pi(u_i) = a$
size of the needed h_a	$k_{u_i} = \alpha_{u_i} \cdot \phi(\mathrm{h}_a)$

Delay Model

cache indicator

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

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

Cloud: $d^c_{u_i}(t) = (1 - x_{\pi(u_i),k}(t)) \cdot (1 - y_{\pi(u_i),j}(t)) \cdot (\frac{k_{u_i}}{v_{c \to m_j}} + \frac{k_{u_i}}{v_{m_j \to u_i}})$

total delay $D_{u_i}(t) = d^r_{u_i}(t) + d^m_{u_i}(t) + d^c_{u_i}(t)$

Cost Model

caching unit cost

cost: $c_r(\mathrm{h}_a) = \gamma_r \cdot \phi(\mathrm{h}_a)$

RSU: $c_{r_k}(t) = \sum_{\mathrm{h}_a \in H} x_{\pi(u_i),k}(t) \cdot c_r(\mathrm{h}_a)$

MBS: $c_{m_j}(t) = \sum_{\mathrm{h}_a \in H} y_{\pi(u_i),j}(t) \cdot c_m(\mathrm{h}_a)$

total cost $c(t) = \sum_{r_k \in R} c_{r_k}(t) + s \sum_{m_j \in M} c_{m_j}(t)$

Model and Formulation

Formulation

objective function

$$\textbf{P1:} \quad \text{minimize } \boldsymbol{D} = \lim_{T\to\infty} \frac{1}{T} \sum_{t\in T} \sum_{u_i \in U} D_{u_i}(t) \qquad (1)$$

$$\text{s.t.} \quad \sum_{\mathrm{h}_a \in H_{r_k}} \phi(\mathrm{h}_a) \le \phi(r_k), \quad \sum_{\mathrm{h}_a \in H_{m_j}} \phi(\mathrm{h}_a) \le \phi(m_j), \qquad (2)$$

$$\sum_{t=1}^{T} c(t) \le C_{max}, \qquad (3)$$

$$x_{\pi(u_i),k}(t),\ y_{\pi(u_i),j}(t) \in \{0,1\} \qquad (4)$$

constraints

Objective: find a caching strategy for contents in *H* to minimize P1 under the constraints (2)-(4).

Model and Formulation

Reformulation

- Perspective: User level → RSU and MBS level
- Method: each RSU r_k as an individual agent
  - caching vector: $\mathbf{x}_k(t) = [x_{1,k}(t), x_{2,k}(t), \cdots, x_{n,k}(t)]^T$, caching decision of r_k.
  - assume $y_{\pi(u_i),j}(t) = 1$ as baseline, and add $\frac{k_{u_i}}{v_{c \to m_j}}$ when $x_{\pi(u_i),k}(t) = y_{\pi(u_i),j}(t) = 0$.

> - Local loss function: $l_k(\mathbf{x}_k(t)) = \sum_{x_a \in \mathbf{x}} x_a(t) \cdot \frac{k_{u_i}}{v_{r_k \to u_i}} + (1 - x_a(t)) \cdot \frac{k_{u_i}}{v_{m_j \to u_i}}$
> - RSU total loss: $\sum_{t=1}^{T} \sum_{k=1}^{|R|} l_k(\mathbf{x}_k(t))$
> - Constraint function: $g(\mathbf{x}) := \sum_{t=1}^{T} \sum_{k=1}^{|R|} c(\mathbf{x}_k(t)) - C_{max} \leq 0$

Model and Formulation

Regret

- To capture the performance of the online caching algorithm

Each RSU:

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

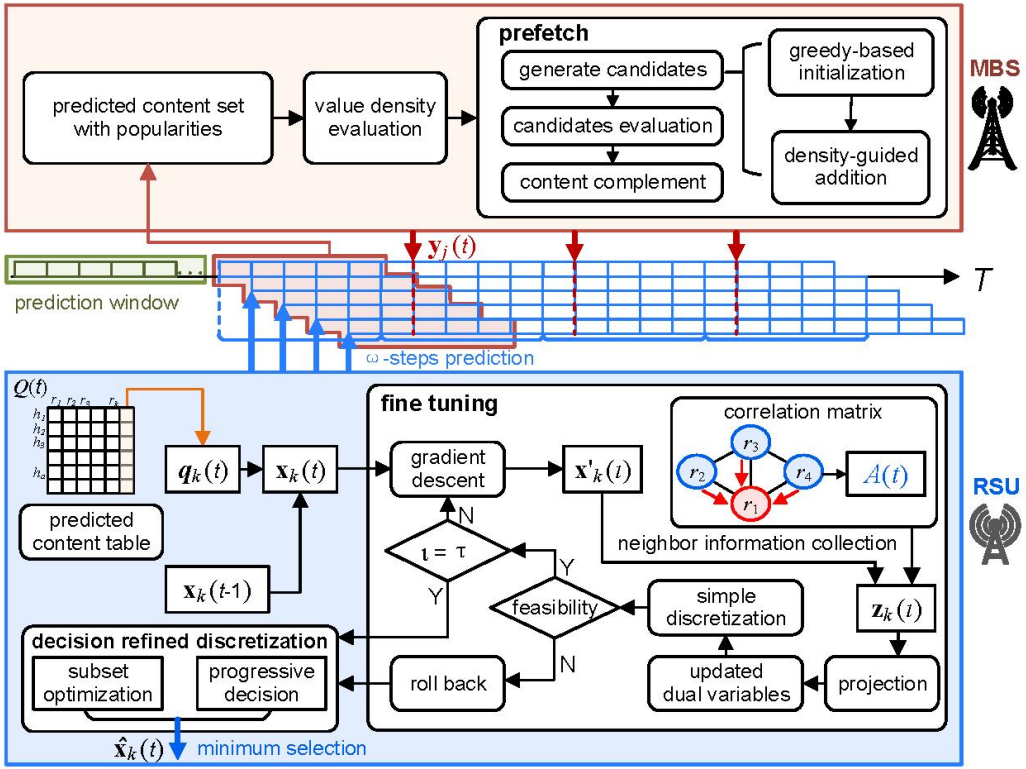
$$\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}\in\mathbf{x}} \sum_{t=1}^{T} \sum_{k=1}^{|R|} l_k(\mathbf{x}_k(t))$$

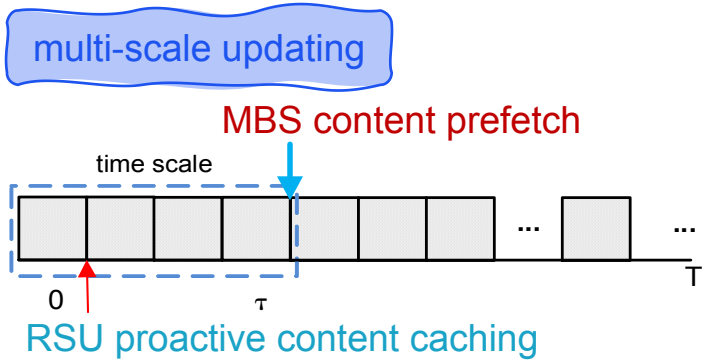
Theorem 1: For each agent $r_k \in R$ and $T \geq 1$, we have

$$\mathbf{reg}(k,T) \leq \frac{1}{2\beta_t}|R| \cdot M + \beta_t \cdot |R| \cdot \overline{l}^2 + \frac{\beta_t \cdot \overline{g}^2 C_{\max}^2 (|R|-1)^2}{|R|^2_{\eta_{t-1}}}$$ with constraint (3).

VOPP framework overview

- **V**ehicular-based **O**nline **P**roactive caching and **P**refetching





MBS: large time scale update

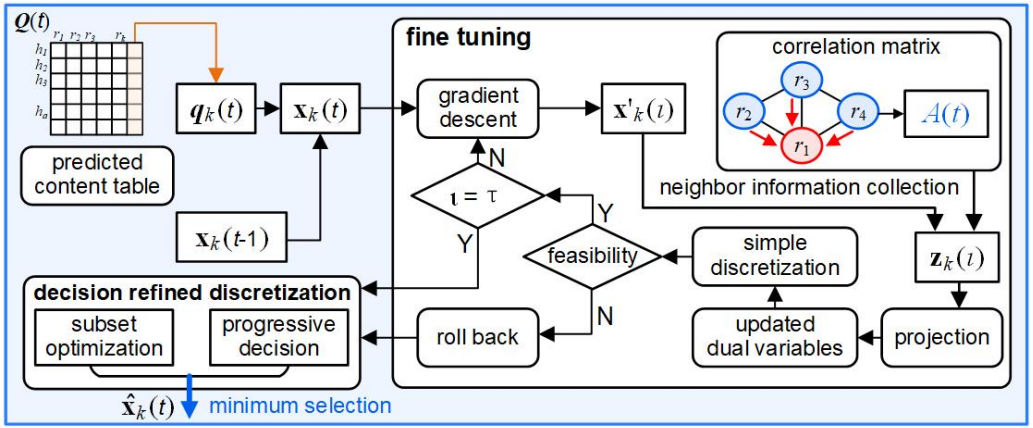
- cover multiple RSUs;
- lower update frequency;

RSU: small time scale update

- distributed near vehicle side;
- high update frequency;

RSU proactive cache

- Each RSU acts as an agent that makes caching decisions based on locally observed requests



RSU

1. user arrival prediction
2. content demand insight
3. gradient descent decision
4. decision refining

RSU proactive cache

- User arrival prediction and user arrival table build

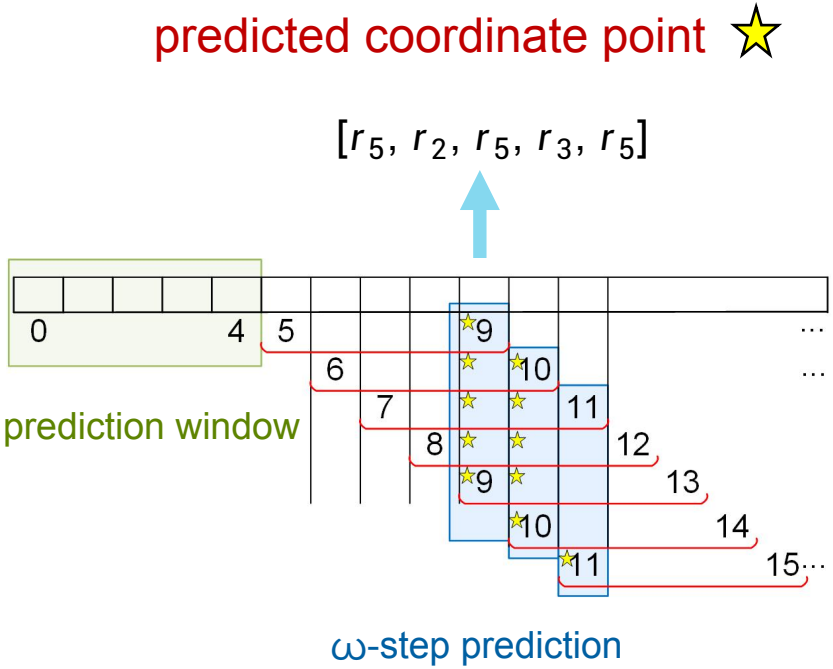
Algorithm 1 Module: Multi-Step User Arrival Prediction

Input: U, R, ω;

Output: Arrival probabilities of users $\mathbf{P}(t)$;

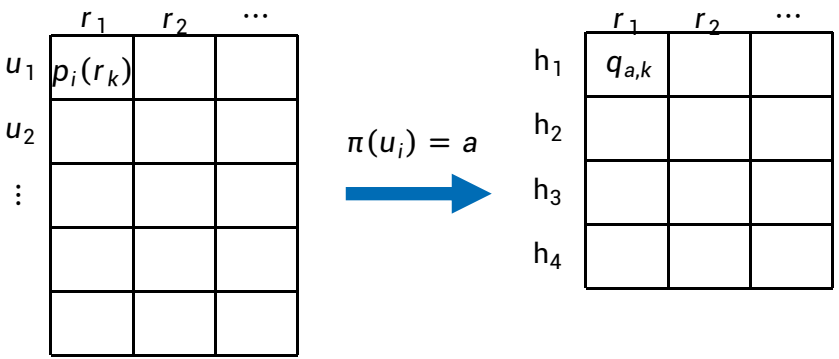
1: **for** each user $u_i \in U$ **do**
2: Predict ω-step future trajectory using social-LSTM;
3: Initialize counter $\rho_i(r_k) = 0, \forall r_k \in R$;
4: **for** each predicted location $\hat{s}_i(t)$ in the ω-step **do**
5: Find all RSUs r_k such that $\hat{s}_i(t) \in \text{coverage}(r_k)$;
6: **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** $\mathbf{P}(t)$;

RSU	r_1	r_2	r_3	r_4	r_5	r_6
$p_i(r_k)$	0	0.2	0.2	0	0.6	0



RSU proactive cache

- Transition from user arrival table to content demand table



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 $\mathbf{Q}(t)$;

- $(1 - p_i(r_k))$: the probability of u_i did not enter RSU r_k.
- $\prod_{u_i \in U, \pi(u_i)=a}(1 - p_i(r_k))$: the probability of h_a did not enter r_k.
- $q_{a,k}(t) \leftarrow 1 - \prod_{u_i \in U, \pi(u_i)=a}(1 - p_i(r_k))$: the probability of h_a enter r_k.

$$q_{a,k}(t) \leftarrow 1 - \prod_{u_i \in U, \pi(u_i)=a} (1 - p_i(r_k))$$

RSU proactive cache

- Decisioning with gradient decent

Algorithm 3 Online Distributed Caching algorithm, (ODC)

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

Large time scale: incrementally update $\mathbf{x}_k(t)$

(12): local loss function $l_k(\mathbf{x}_k(\iota))$ — local gradient

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

penalty term: dual variable, constraint violation

local gradient: step toward lower loss

penalty term: step away from constraint violation

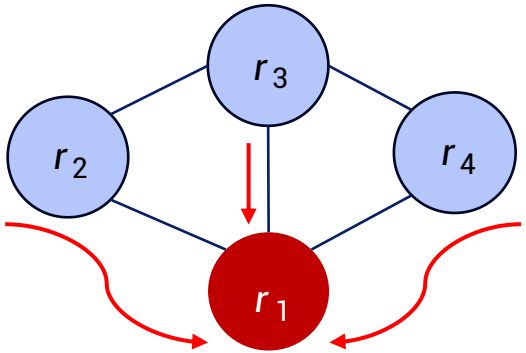
RSU proactive cache

Algorithm 3 Online Distributed Caching algorithm, (ODC)

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

Decisioning with gradient decent

- row-stochastic influence matrix $[A(t)]_{kj}$ reflects how previous neighbor information influence current decision.



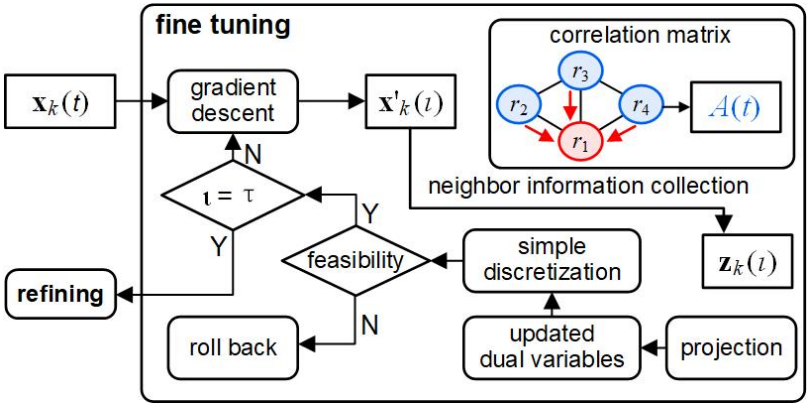
The intermediate decision vector $\mathbf{z}_k(\iota)$ aggregates the decisions of its neighboring RSUs by weights from $A(t)$.

RSU proactive cache

Algorithm 3 Online Distributed Caching algorithm, (ODC)

Require: $\mathbf{Q}(t)$, α, τ, $A(t)$;
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 τ slices; ▷ fine-grained slices
5: **for** $\imath = 0$ to τ **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** $\mathcal{X} = \{\hat{\mathbf{x}}_k(t) | t \in \{1, \ldots, T\}\}$;

- Decisioning with gradient decent



Limiting out-of-bound values

Update dual variable

Discretize vectors via thresholding

Feasibility check with roll back mechanism

Decision refining

RSU proactive cache

Decision refining

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

$\mathbf{x}_k(t) = \mathbf{0}$

$\mathbf{x}_k(t) = [0100]^T$

continuous	0.31	0.28	0.53	0.44	0.72	0.61
discretized	0	0	1	0	1	1

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** $\hat{\mathbf{x}}_k$.

storage constraint

DP object size: content size $\phi(h_a)$

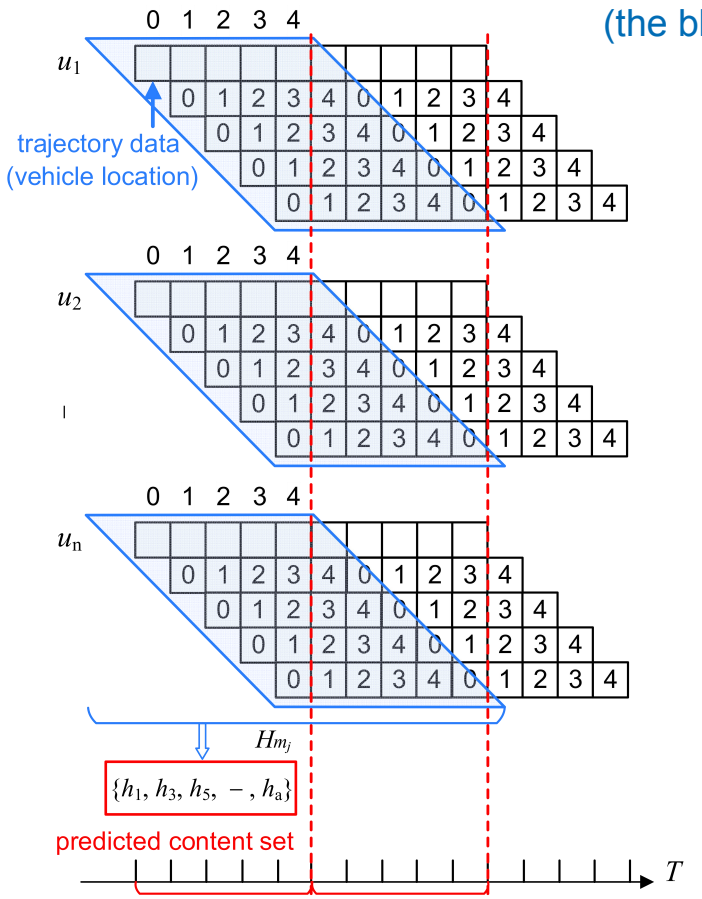
DP object value: $\Phi(h_a, r_k)$ under the predicted user demands

greedy solution

DP solution

choose the lower $l_k(\mathbf{x})$

MBS prefetch



predicted trajectory window (the blue area parallelogram) → associate each coordinate to MBS → content set H_{m_j}

the popularity $\mathcal{N}(h_a)$ of each content h_a was accumulated from the predicted content set H_{m_j}

Definition 2 (delay reduction): the delay benefit when the content is cached at MBS instead of cloud:

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

$D_{u_i}(t)|_{x_{a,k}=0, y_{a,j}=0}$ $D_{u_i}(t)|_{x_{a,k}=0, y_{a,j}=1}$

Definition 3 (value density): the caching worthiness of the content h_a on a given MBS m_j:

$$V_j(h_a) = \mathcal{N}(h_a) \cdot \Delta(h_a)/\phi(h_a)$$

popularity delay reduction requiring storage

MBS prefetch

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 $H_{m_j}^{\text{top}(k)} = \{h_a | \arg\max_{h_a \in H}^{k} V_j(h_a)\}$;
7: $\hat{H}_{m_j} \leftarrow H_{m_j} \cup H_{m_j}^{\text{top}(k)}$;
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** $\mathbf{y}_j(t)$

- Greedy selection: fill the MBS storage with high value contents
- Top-*k* value selection: explicitly select high value contents regardless storage limit
- Supplementary selection: fill the remaining MBS storage based on delay reduction $\Delta(h_a)$

greedy selection

top-*k* value selection

candidate set $\hat{H}_{m_j}$

local DP optimize, considering only h_a in $\hat{H}_{m_j}$

pre-decision with DP results

supplementary selection: in case the unpredicted arriving vehicles

CONTENTS

Experiment and Results

Basic Setting

- Hardware: Windows 10 with an Intel(R) Xeon(R) Silver 4210R CPU @ 2.40GHz, NVIDIA RTX5000 GPU, and 32GB memory.
- Dataset: GPS traces from 10,357 taxis (Microsoft).
- Center point: Financial Street, Beijing ([116.36032115, 39.911045075]).
- Range: 2.9 km^2 square area.

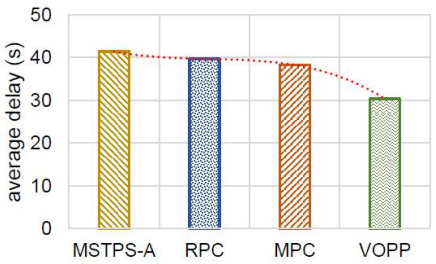
  longitude range from 116.3518143 to 116.368828;

  latitude range from 39.8996048 to 39.92248535;

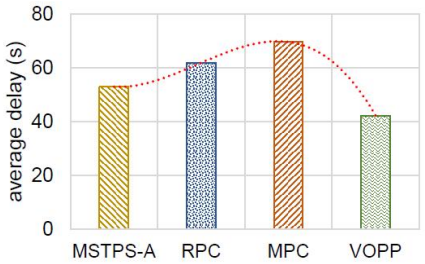
Experiment and Results

❑ Evaluation of Average Delay

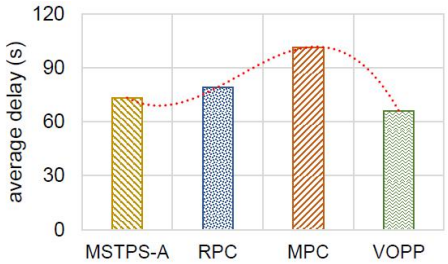
- VOPP achieves the lowest average transmission delay.
- VOPP maintains high performance with an average delay of 30.34 seconds.
- VOPP maintains low delay without performance degradation.



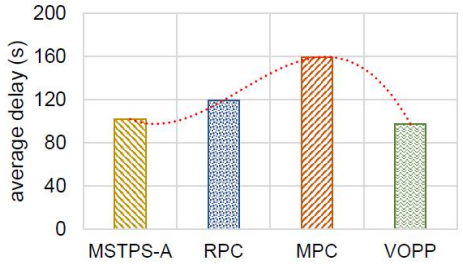
(a) # of users 100.



(a) # of users 150.



(a) # of users 200.



(a) # of users 264.

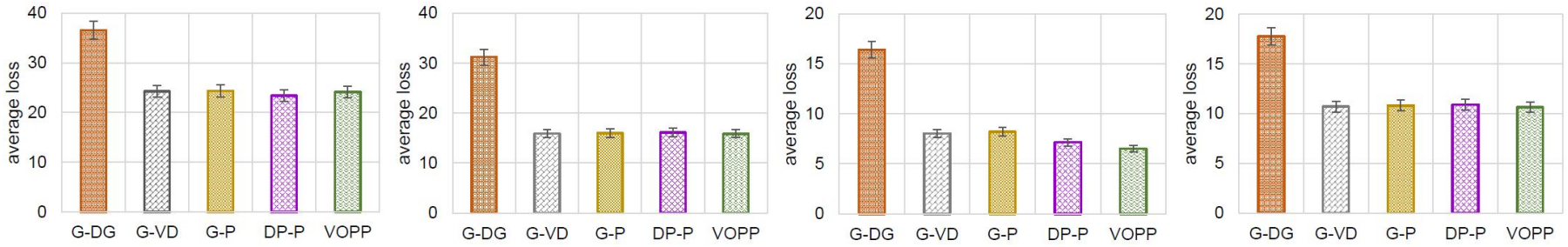
Fig. 1. obj. (average latency) for different numbers of users.

[1] 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).

Experiment and Results

MBS-layer Evaluation

- VOPP consistently reduces transmission delay as MBS storage increases.
- VOPP achieves more effective caching decisions.
- VOPP maintains low average delay across diverse MBS storage capacities.



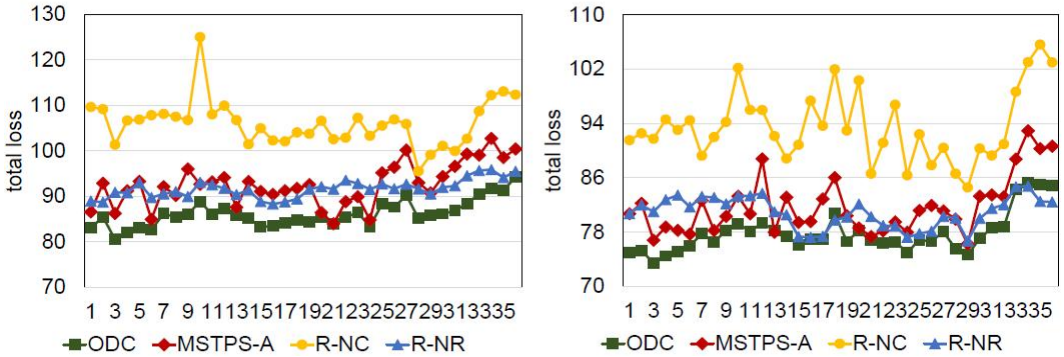
(a) $\phi(m_j)$ = 5G. (b) $\phi(m_j)$ = 6G. (c) $\phi(m_j)$ = 7G. (d) $\phi(m_j)$ = 8G.

Fig. 2. obj. (average latency) for different sizes of MBSs.

Experiment and Results

RSU-layer Evaluation

- VOPP consistently achieves low delay.
- VOPP ensures effective caching by refining decisions based on content priorities.
- VOPP improves caching efficiency and responsiveness to user mobility.

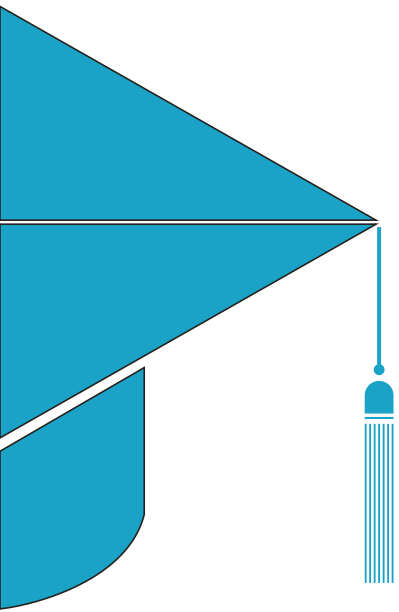


(a) normal (1), large (3). (b) normal (2), large (4).

Fig. 3. obj. (average latency) for different sizes of RSUs.

Conclusion

- **VOPP**: a hierarchical framework jointly optimizes RSU-level proactive caching and MBS-level prefetching decisions.
- **RSU-level**: a distributed online convex optimization model with coordination among agents and a decision enhancement mechanism.
- **MBS-level**: a predicted content set and employ a hybrid greedy-complementary algorithm to generate robust prefetching decisions.
- **Experiments:** GPS traces show consistent delay reduction across diverse user densities and mobility patterns.
- **Future work:**
  - learning-based demand prediction
  - federated multi-domain extension
  - privacy-aware caching strategies



THANK YOU