北京工業大學

BEIJINGUNIVERSITYOFTECHNOLOGY

QoS-aware Dynamic Service Caching and Updating in Cost-efficient Multi-Access Edge Computing

Shuaibing Lu, **Xin Jin**, Jie Wu, Shuyang Zhou, Shen Wu, Jackson Yang, Ran Yan

Contents

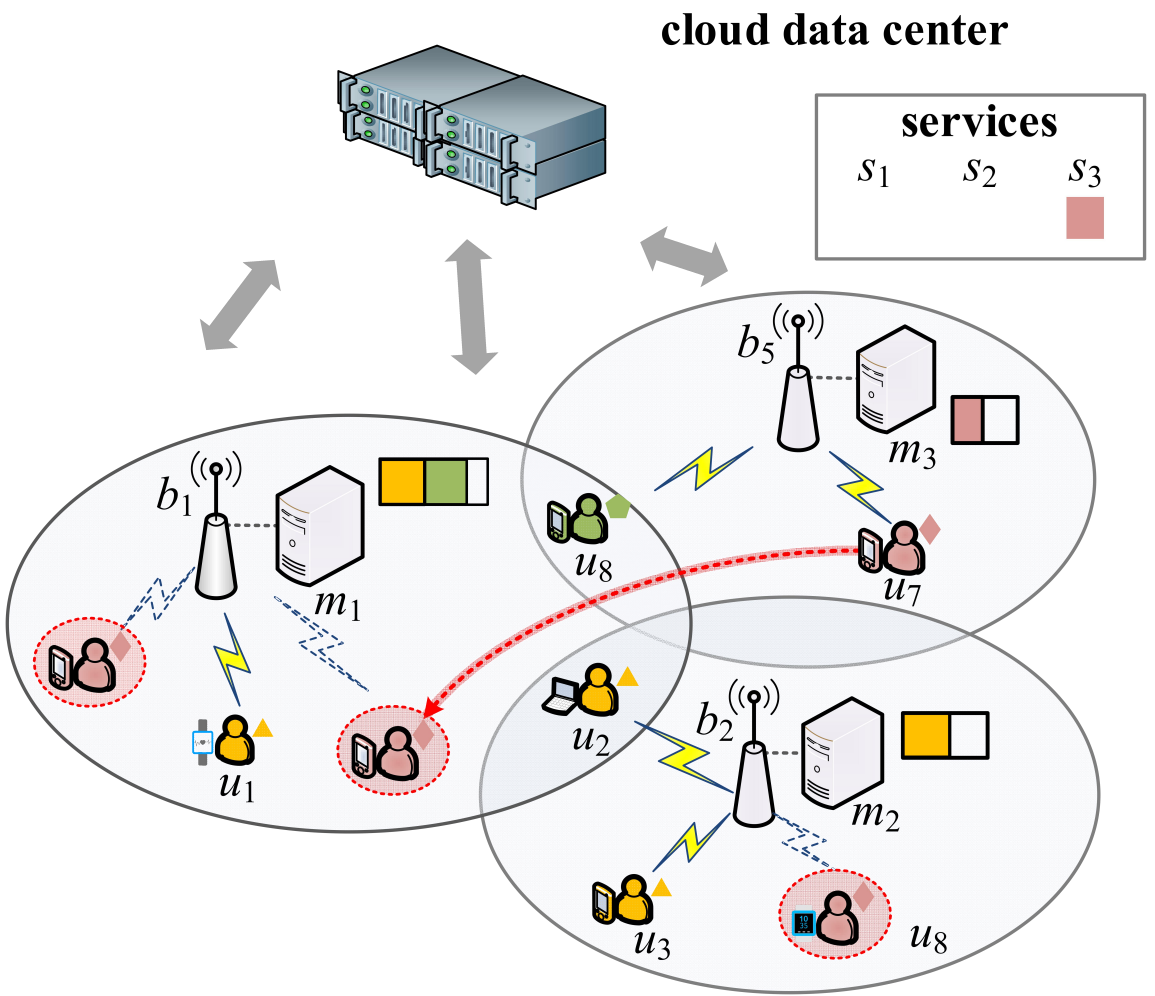
Part 1

Background and Motivation

Background and Motivation

➢ Multi-Access Edge Computing

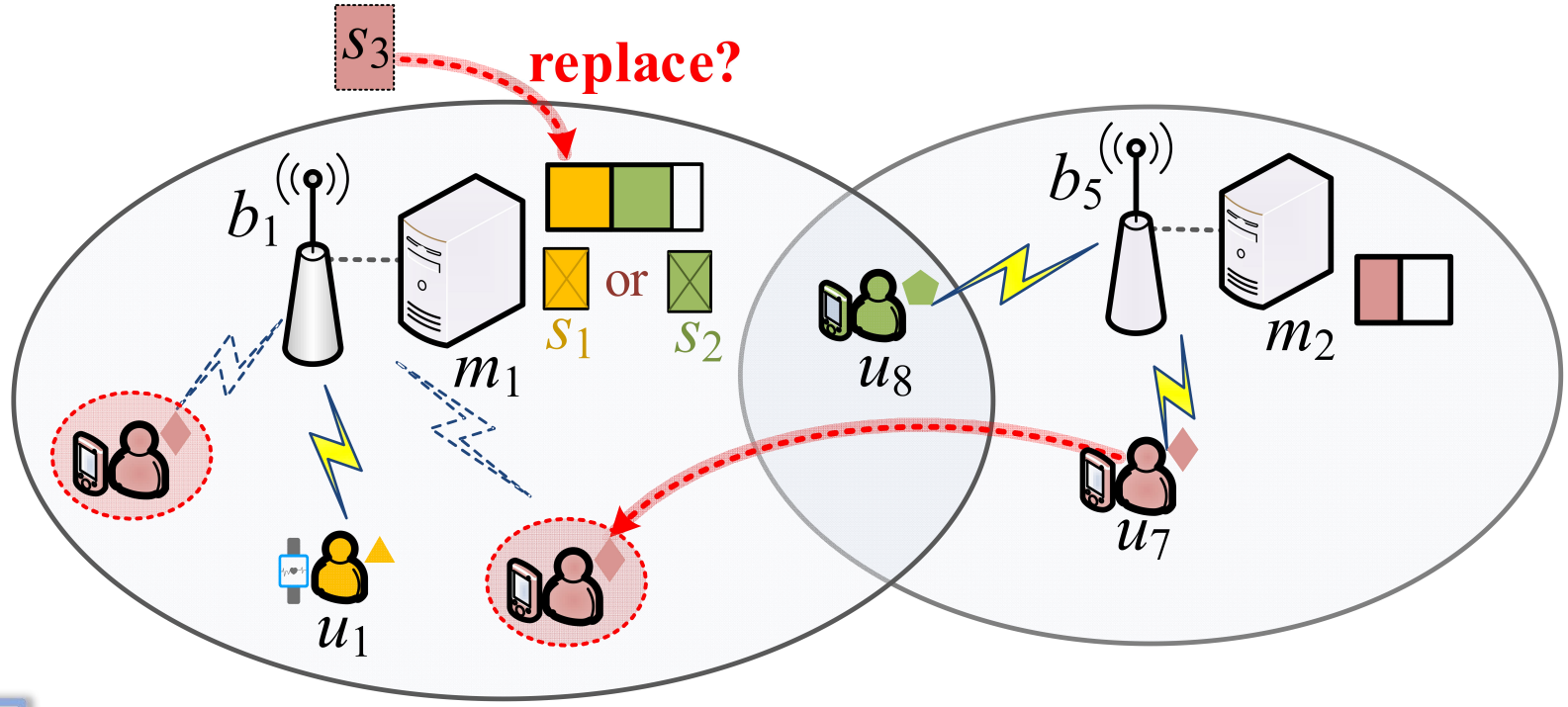
- To address the increasing demand for real-time data and services
- Deploy edge servers at the network edge
- Bring computation and storage closer to end-users and devices
- Low-latency and high-bandwidth service delivery



Background and Motivation

➢ Why do we need dynamic caching and updating?

- Users mobility
- New requests
- Low cost and latency
- Limited storage capacity



➢ Key Problems and Objective

◆ Key Problems:

- which services to cache
- Whether add replications
- How to obtain replications
- How to update services

◆ Objective:

- Reduce delay of users and system cost under the condition of limited capacity

Part 2

Model and Formulation

Model and formulation

北京工業大學 BEIJINGUNIVERSITYOFTECHNOLOGY

➢ System Model

$$\mathbf{U} = \{u_i\}, \mathbf{S} = \{s_j\}, \mathbf{M} = \{m_k\}$$

$$u_i = (q_{u_i}, z_{u_i}^{cal}, \kappa_{u_i})$$

$$s_j = (c_{s_j}, \varphi_{s_j})$$

$$m_k = (R_{m_k}^{s}, R_{m_k}^{cal})$$

➢ QoS Model

$$\boldsymbol{D} = \sum_{u_i \in \boldsymbol{U}} (d_{u_i}^{cal} + d_{u_i}^{comm})$$

$$d_{u_i}^{cal(m_k)} = \frac{z_{u_i}^{cal}}{R_{m_k}^{cal}} \qquad d_{u_i}^{cal(\boldsymbol{c})} = \frac{z_{u_i}^{cal}}{R_{\boldsymbol{c}}^{cal}}$$

$$d_{u_i}^{comm(m_k)} = l_{m_k,m_{k'}} \cdot \frac{z_{u_i}^{cal}}{\lambda_{m_k,m_{k'}}}$$

$$d_{u_i}^{comm(\boldsymbol{c})} = l_{\boldsymbol{c}} \cdot \frac{z_{u_i}^{cal}}{\eta \cdot b_{\boldsymbol{c}}}$$

Model and Formulation

➢ Cost Model

$$C = \sum_{s_j \in \mathbf{S}} \left(C_{s_j}^{stor} + C_{s_j}^{migr} \right)$$

$$C_{s_j}^{stor} = N_{s_j} \cdot \varphi_{s_j}$$

$$C_{s_j}^{migr(m_k, m_{k'})} = \gamma_{\boldsymbol{m}} \cdot c_{s_j} \cdot l_{m_k, m_{k'}}$$

$$C_{s_j}^{migr(m_k, \boldsymbol{c})} = \gamma_{\boldsymbol{c}} \cdot c_{s_j} \cdot l_{\boldsymbol{c}}$$

➢ Formulation

$$\mathbf{P}_1 : \min_{x(m_k, s_j)} \{\mathbf{D}, \mathbf{C}\}$$

$$\text{s.t.} \sum_{s_j \in \mathbf{S}_{m_k}} c_{s_j} \leq R_{m_k}^s, \quad \forall m_k \in \mathbf{M}$$

$$x(m_k, s_j) \in \{0, 1\}, \quad \forall m_k \in \mathbf{M}, \forall s_j \in \mathbf{S}$$

$$\mathbf{P}_2 : \min_{x(m_k, s_j)} \{\alpha \mathbf{D} + \beta \mathbf{C}\}$$

$$\text{s.t.}(2)(3)$$

Part 3

Algorithm Design

Algorithm Design

Three Stages —address→ **Key Problems**

- Initial service caching
  - which services to cache
- Service updating decision-making
  - Whether add replications
  - How to obtain replications
- Service updating replacement
  - How to update services

Algorithm Design

➢ Initial service caching

Service placing problem

Transform

knapsack problem

User oriented Service Caching (USC) Algorithm

Initial service caching strategy

- Construct the value table for each server
- Calculate service value of services requested at the edge server
- Update the table according to the recursion relationship
- Trace back the table after finalizing the table of edge servers
- Obtain the caching strategy based on the backtracking results

Algorithm Design

➢ Service updating decision-making

New requests

↓

Preliminary selection → • Set an appropriate threshold, only requests above the threshold can proceed to further selection

↓

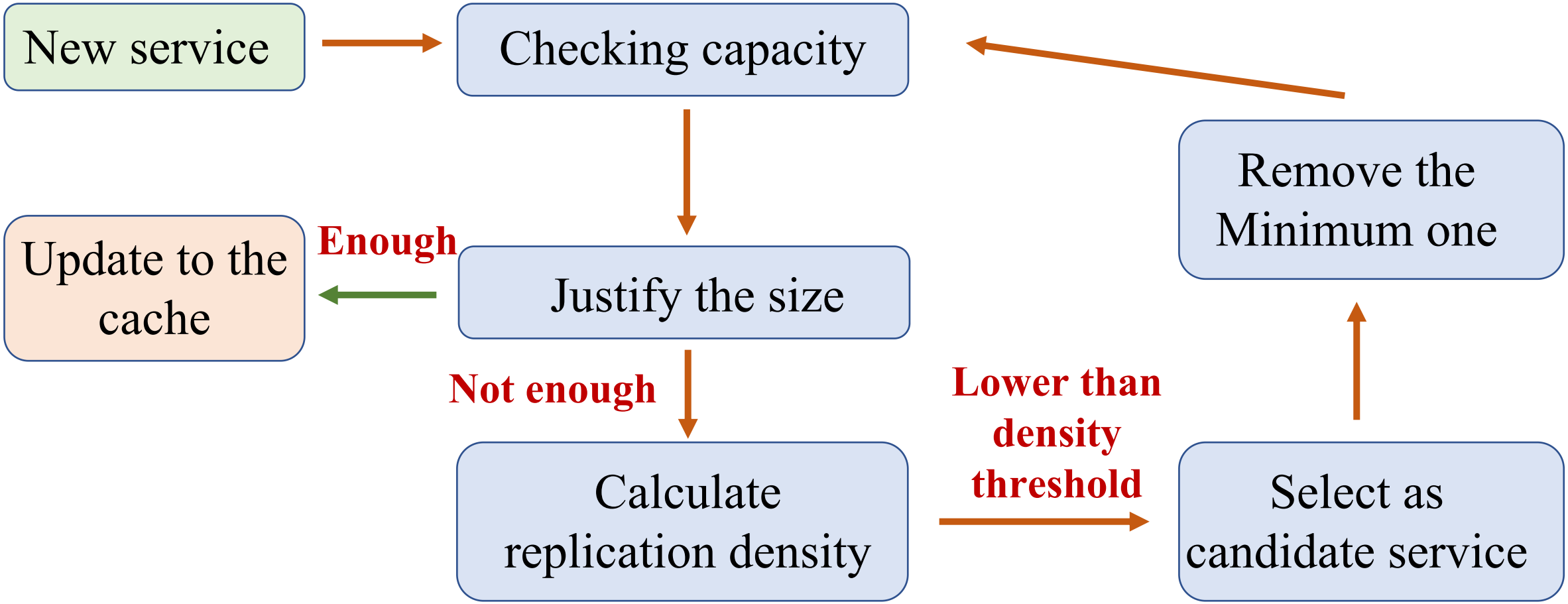
Further selection → • Using SUD algorithm based on Q-learning to deal requests

↓

Add replications → • Based on the decision-making strategy, add replications for selected services.

Algorithm Design

➢ Service updating replacement



Part 4

Experiment

Experiment

➢ Basic Setting

- Hardware: Windows 11, i9-12900KF CPU, NVIDIA RTX3080 GPU, 32GB memory
- Dataset: China Telecom Shanghai Company (3233 base station locations and corresponding user connections in June 2014)
- Scales:

  Group 1: 10 servers, 15 services, 40 users

  Group 2: 20 servers, 30 services, 80 users

  Group 3: 30 servers, 45 services, 120 users

Experiment

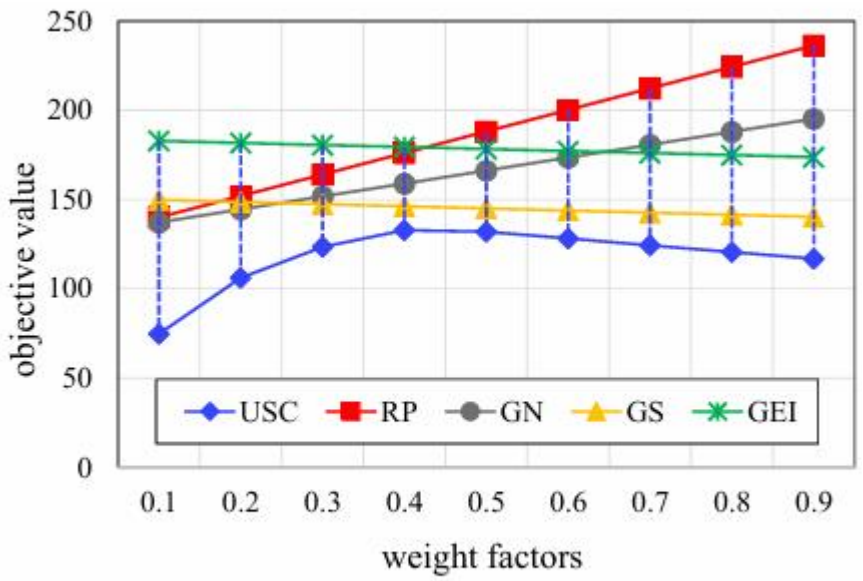
➢ Initial service caching

- Greedy based on the Number of users for the services (GN) algorithm
- Greedy based on the Size of services (GS) algorithm
- Greedy based on the Evaluation Indicator of services (GEI) algorithm
- Random Placement (RP) algorithm
- **User oriented Service Caching (USC) Algorithm**

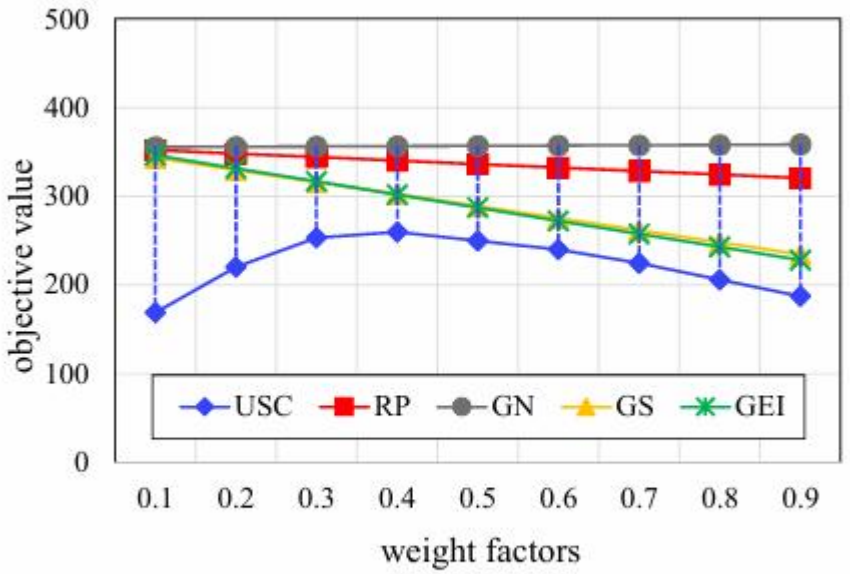
Experiment

➢ Initial service caching

In all three groups of experiments, the USC algorithm consistently maintained the lowest objective value under all weight settings.



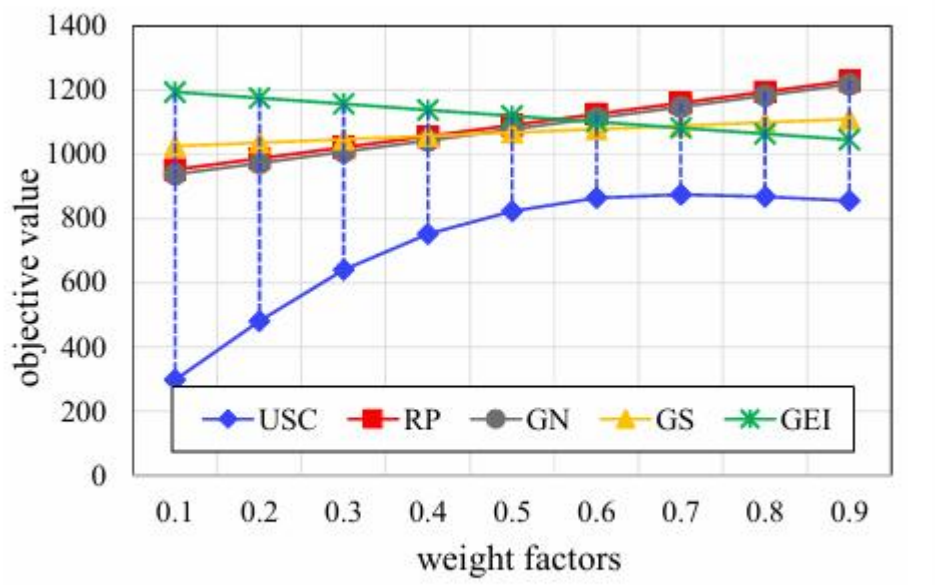
(a) Group 1.



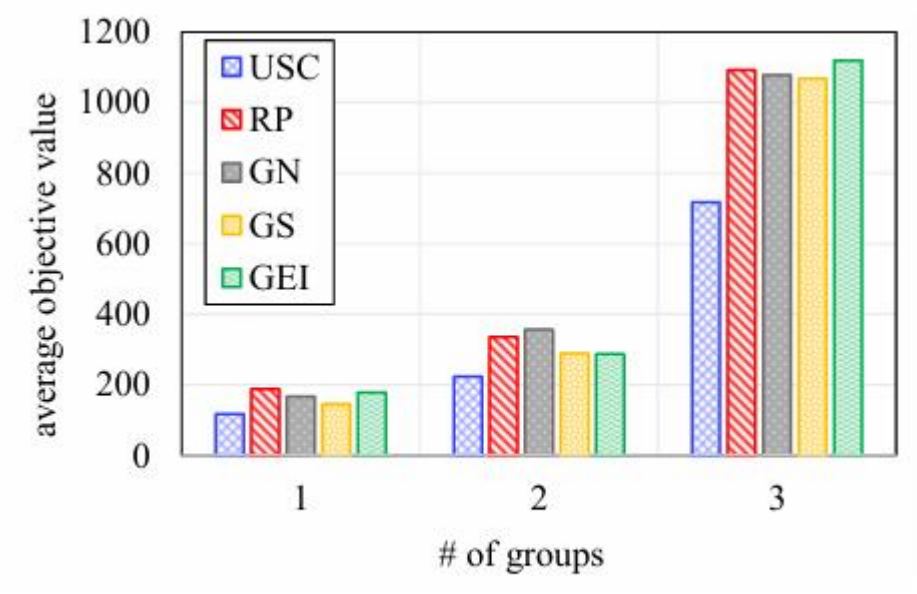
(b) Group 2.

Experiment

➢ Initial service caching



(c) Group 3.



(d) Average

Compared to other algorithms, the superiority of the USC algorithm gradually becomes apparent in the average results across the 10 sets of weights from scale 1 to scale 3

Experiment

➢ Service updating decision-making

- Service Updating from the Cloud (SU-C) algorithm
- Service updating from the Edge or Cloud (SU-EC) algorithm
- **Service Updating Decision-making based on the QoS (SUD) Algorithm**

Experiment

➢ Service updating decision-making

TABLE I

COMPARISON OF DECISION-MAKING ALGORITHMS.

Group	SUD	SU-C	SU-EC
1	48.88 ± **8.07**	98.40 ± **18.18**	92.20 ± **11.72**
2	194.66 ± **55.08**	327.14 ± **141.81**	263.31 ± **81.80**
3	493.06 ± **19.16**	859.05 ± **58.37**	577.57 ± **182.44**

The SUD algorithm achieves the best result with the lowest objective value and the lowest variance in three groups

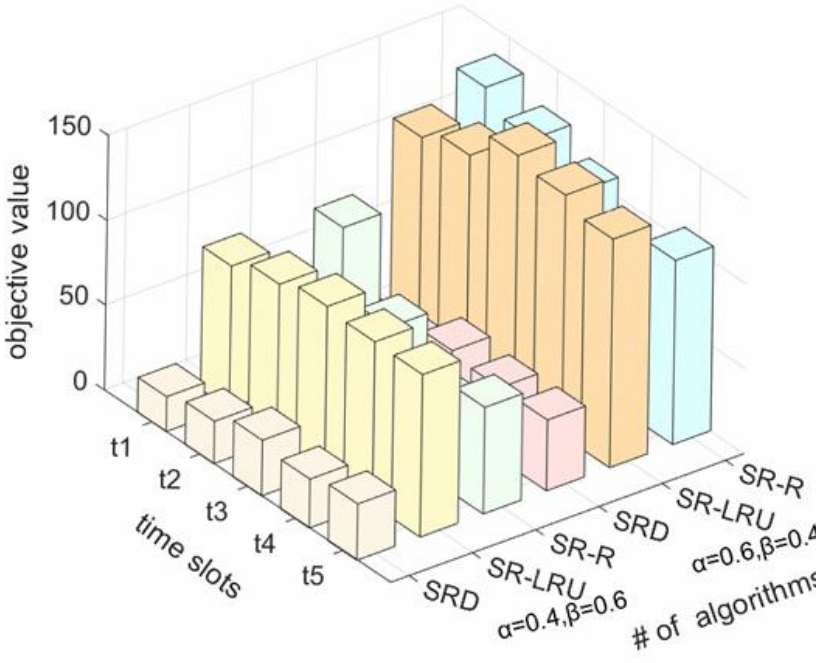
Experiment

➢ Service updating replacement

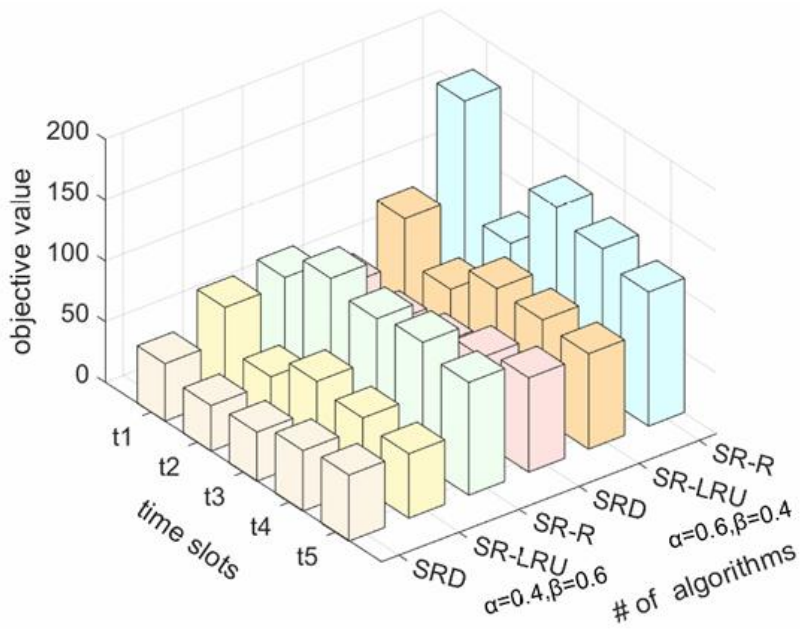
- Service Replacement based on LRU (SR-LRU) algorithm
- Service Replacement Randomly (SR-R) algorithm
- **Service Replacement based on the distribution of services (SRD) Algorithm**

Experiment

➢ Service updating replacement



(a) group 1.



(b) group 2.

The SR-LRU and the SR-R exhibit unstable performance under the influence of different weight settings, whereas the SRD algorithm consistently achieves the lowest value under both weight settings

Part 5

Conclusion

Conclusion

This paper focuses on addressing the service caching problem in a dynamic and resource-constrained multi-access edge computing environment.

➢ Contributions

- We design the QoS model and the cost model for user latency and system cost.
- We develop three algorithms to solve the problem in three phases of service caching and updating.
- We conduct extensive experiments to compare our algorithms with several baselines.

北京工業大學

BEIJINGUNIVERSITYOFTECHNOLOGY

THANK YOU