This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2025.3531967

Cost-Effective Edge Data Caching with Failure Tolerance and Popularity Awareness

Ruikun Luo, Zujia Zhang, Qiang He, *Senior Member, IEEE,* Mengxi Xu, Feifei Chen, *Member, IEEE,* Xiaohai Dai, Song Wu, *Member, IEEE,* and Hai Jin, *Fellow, IEEE*

Abstract—In the mobile edge computing environment, caching data in edge storage systems can significantly reduce data retrieval latency for users while saving the costs incurred by cloud-edge data transmissions for app vendors. Existing *edge data caching* (EDC) methods prioritize popular data and aim to minimize users' data retrieval latency and system storage costs jointly. However, these EDC methods often rely on the assumption that data popularity always follows certain distributions. As a result, they cannot properly adapt to the fluctuations in data popularity due to user mobility or unexpected increases in user demands. Meanwhile, unlike cloud data centers, complex and fragile edge servers are more likely to experience physical failures or network outages, presenting new challenges for EDC strategies. Specifically, when an edge server fails or experiences an outage, cached data may become temporarily unavailable, leading to increased latency as requests are redirected to alternative servers or the cloud. In this paper, to enable *uncertainty-aware edge data caching* (uEDC), we first model the problem as a robust optimization problem and propose an optimal algorithm named uEDC-B to find the optimal uEDC solution. To address the high computational complexity of uEDC-B, we introduce an approximate algorithm named uEDC-L based on linear decision rules. Theoretical analysis and extensive experiments on a real-world dataset demonstrate that the proposed methods outperform two state-of-the-art approaches in handling the uncertainties in data popularity and edge server failure with a significant performance improvement of 59.27% in data retrieval latency and 55.07% in data caching cost.

Index Terms—mobile edge computing, edge data caching, data popularity, optimization approach

1 Introduction

In the digital era, mobile edge computing, a new distributed computing paradigm, is rapidly shifting the resource provisioning from reliance on centralized cloud data centers for data processing and storage to a more distributed approach [1]–[3]. By relocating data and computational tasks from central data centers to edge devices or edge servers near users, mobile edge computing reduces data processing latency and decreases the burden on data centers significantly [4]–[6]. Comprised of connected edge servers in a specific geographic area, an *edge storage system* (ESS) is capable of delivering data to users in the area with low latency [7], [8]. In this context, *edge data caching* (EDC) has become a critical technique for enhancing data access efficiency and reducing network congestion.

App vendors like TikTok and YouTube can reduce users' data retrieval latency effectively by caching popular videos on ESSs [9], [10]. They can also reduce the costs associated with data transmissions from cloud data centers to users since transferring data between edge servers (edge-to-edge) is much cheaper than between the cloud and edge servers (cloud-to-edge) [11]. For example, Amazon Web Services charges \$0.05-\$0.09 for every GB of data transferred out of its S3 to the internet [12], while only \$0.01 for transferring every GB of data between edge servers within the same AWS Wavelength Zone at the network edge [13]. Edge data caching also enables real-time decision-making processes for many applications in dynamic urban environments. For example, in smart cities, traffic management systems may rely on edge caching to process large volumes of traffic data in real time, enabling swift adjustments to traffic flow.

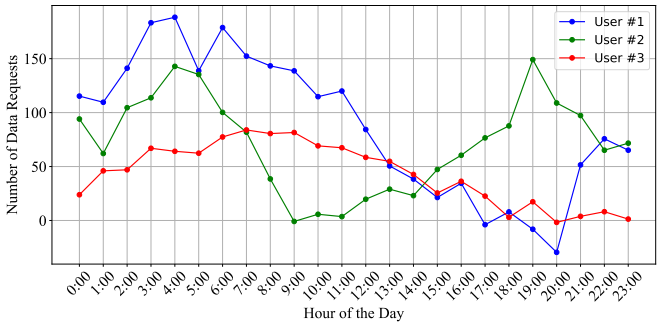


Fig. 1: Data requests over time. We use Apache Kafka to conduct a statistical analysis of the average number of access requests for a specific type of data, such as video streaming here, from users across four distinct geographical regions within a single day.

In mobile edge computing environments, edge servers are commonly deployed at 5G/6G base stations near users [14]. Edge servers in an area communicate with each other via high-speed links [7], [15]. Unlike cloud servers, the limited physical sizes of edge servers dictate their constrained resources [16]. Their constrained storage capacities,

- *R. Luo, Z. Zhang, Q. He, M. Xu, S. Wu, and H. Jin are with National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, China. Email: {rkluo, heliotrope, hqiang, u202112027, xhdai, wusong, hjin}@hust.edu.cn.*
- *F. Chen is with the School of Information Technology, Deakin University, Geelong, VIC 3125, Australia. Email: feifei.chen@deakin.edu.au.*

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2025.3531967

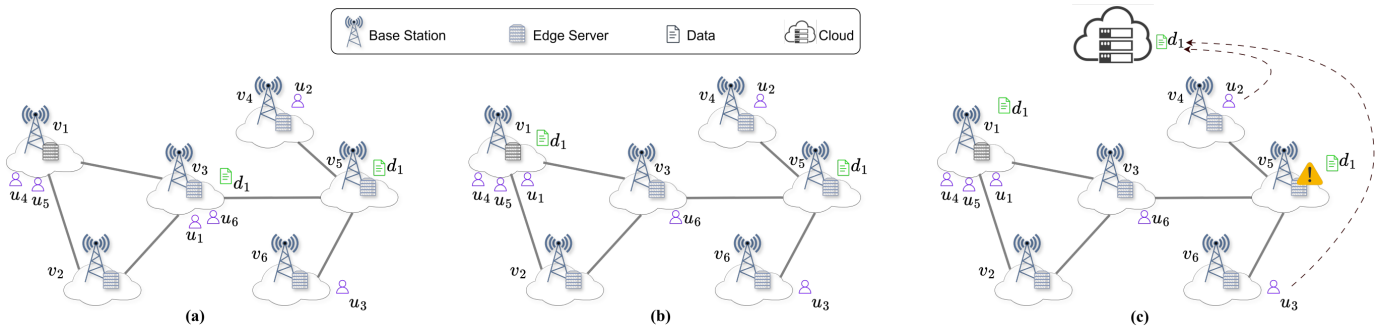


Fig. 2: Edge data caching strategy under popularity uncertainty and edge server failures. The popularity uncertainty is represented by the mobility of user u_1, while the uncertainty of server failures includes the failure of v_5.

in particular, impact the system's ability to accommodate users' data demands. To optimize the utilization of limited edge storage resources, current edge data caching methods typically cache the most popular data, i.e., data requested by many users [17]–[19]. Based on data popularity, such methods rely on accurate data popularity predictions or assumptions regarding the distribution of data requests. However, as shown in Fig. 1, fluctuations in the volume of edge data access requests in the mobile edge computing environments are common and notoriously challenging to predict [20]. These fluctuations may be attributed to various factors, including user mobility, evolving user behavior patterns, etc. These pose profound challenges for edge data caching. For instance, breaking news on social media can rapidly increase the popularity of relevant video content, potentially rendering existing caching strategies ineffective. Failing to respond to these swift changes in data demands. Edge caching systems might end up storing infrequently-accessed data. Users have to fetch viral videos from remote cloud servers, which leads to increased data retrieval latency. Take Fig. 2 as an example. For easy description, we assume that all users $\{u_1, \ldots, u_6\}$ request data d_1. Fig. 2(a) shows a data popularity-based EDC strategy that caches data d_1 on edge servers v_3 and v_5 to ensure all users can retrieve d_1 within one hop in the system. However, due to user mobility, u_1 served by v_3 moves into the coverage of v_1. A better EDC strategy, as shown in Fig. 2(b), is to cache d_1 on v_1 to serve the most users demanding d_1. Overlooking the uncertainties in data requests can undermine the fundamental objectives of edge caching, i.e., to bring popular data closer to users and reduce their data retrieval latency.

Meanwhile, unlike cloud data centers typically housed in well-maintained and protected environments, edge servers are often deployed in more varied and potentially less secure or friendly locations, making them more susceptible to failures [21], [22]. This is primarily due to their proximity to end-users for lower latency, necessitating their placement in diverse environments that might not offer the same infrastructure robustness and physical security as traditional data centers [23]. Consequently, edge servers are exposed to a higher risk of physical damage, environmental hazards, power instability, and network issues [24], [25]. These factors collectively contribute to more server failures at the edge than their counterparts in cloud data centers [26]. When an edge server fails, the data cached on the failed edge server becomes inaccessible and users' data requests will be rerouted to other edge servers or cloud. This increases the data retrieval latency and impacts their quality of experience. Take Fig. 2(b) as an example. An EDC method without considering edge server failures may cache data d_1 on edge servers v_1 and v_5. If v_5 fails, as shown in Fig. 2(c), this EDC strategy is not robust because users u_2 and u_3 cannot retrieve d_1 within one hop in the ESS. Their data requests have to be offloaded to the remote cloud, which increases their data retrieval latency significantly. A more robust strategy that accounts for server failures might involve caching the data on v_1, v_4, and v_6.

To enable robust edge data caching, this paper makes the first attempt to investigate the *uncertainty-aware edge data caching* (uEDC) problem. Its main contributions can be summarized as follows:

- We model and formulate the uEDC problem, and prove its $\mathcal{NP}$-hardness.
- We propose two approaches to solve the uEDC problem, i.e., uEDC-B and uEDC-L, the former for finding optimal robust EDC strategies and uEDC-L for finding approximate uEDC solutions in large-scale scenarios.
- We prove the theoretical performance of uEDC-L and conduct extensive experiments to evaluate the performance of uEDC-B and uEDC-L on a widely-used real-world dataset.

This paper is organized as follows. We formulate the uEDC problem and prove that it is $\mathcal{NP}$-hard in Section 2. In Section 3 and Section 4, we introduce uEDC-B and uEDC-L in detail and evaluate uEDC-B and uEDC-L experimentally in Section 5. Section 6 reviews the related work. Finally, Section 7 summarizes this paper and points out the future direction.

2 Problem Formulation

In this section, we formulate the uEDC problem and then prove its hardness theoretically. Table 1 summarizes the main notations used in this paper and their descriptions.

2.1 System Model

Given an edge storage system deployed in a specific geographic area, let V represent the set of edge servers in the system and $D = \{d_1, ..., d_n\}$ represent users' data requests. In the mobile edge computing environment, the end-to-end

TABLE 1: Summary of notations

Notation	Description
$a_{i,k}$	binary variable indicating cache d_i on edge server v_k
C_k	available cache space on v_k
d_i	data i
$m_{i,j}$	satisfiable requests from v_j at d_i
$p_{j,k}$	delay between v_j and v_k
$q_{i,j}$	unsatisfiable requests from v_j at d_i
$\mathcal{R}$	request uncertainty set
R	set of data requests
$r_{i,j}$	total requests from v_j at d_i
s_k	unit cache cost on v_k
t_j	latency between v_j and the cloud
V	set of edge servers
v_j	edge server j
$\mathcal{W}$	edge server failure uncertainty set
w_k	binary variable denotes whether v_k is fail

data retrieval latency consists of the latency between the user and the edge server serving the user and the latency in data transmission between servers. Similar to previous studies [15], [27], this study focuses on the latter because the former is not impacted by the EDC strategy. Let us assume that an edge server v_j needs to fetch data d from v_k to serve a user. The overall data retrieval latency is the latency between v_j and v_k. Otherwise, the data is transmitted from the cloud to the user with a latency of t_j. The requests for different data d_i vary for users and are aggregated onto edge servers. Thus, different edge servers usually have different data needs within the system. Let $r_{i,j}$ denote edge server v_j's requests for data d_i. The number of data requests for data d_i determines its popularity. Caching popular data on edge storage systems is critical for cost-effectiveness because it can minimize the data retrieval latency for as many users as possible. Finding an EDC strategy for data d_i is to find out whether to cache d_i on each of the edge servers in the system. It can be represented by an array of binary variables $a_{i,j}$. If $a_{i,j} = 1$, it indicates that data d_i is cached on edge server v_j. In this context, the data retrieval latency can be modeled as follows:

$$\mathcal{L} = \sum_i \sum_j r_{i,j} p_{j,k} \tag{1}$$

where $p_{j,k}$ denotes the minimum latency between edge server v_j and v_k and edge server v_k is satisfies $a_{i,k} = 1$.

Due to differences in hardware configurations, geographic locations, remaining resources, etc., the cost of caching data may vary across different edge servers [28]. Let s_k represent the cost of caching a data unit on edge server v_j. The cost of caching $\mathcal{C}$ in the system can be calculated as follows:

$$\mathcal{C} = \sum_k \sum_i s_k a_{i,k} \tag{2}$$

As mentioned in Section 1, edge servers often suffer from constrained storage resources. Thus, the number of data cached on each edge server v_k cannot violate the available server capacity constraint:

$$\sum_i a_{i,k} \leq C_k \tag{3}$$

The key optimization objective is to serve the maximum users' data requests with the minimum data caching costs. However, simply pursuing the minimization of caching costs usually results in high data retrieval latency. For instance, if a data item is cached on only one edge server within the entire system, the storage cost will be minimal, but users' average data retrieval latency will be high because distant users have no other choice but to retrieve data from that edge server. Therefore, edge data caching should optimize the caching cost and data retrieval latency jointly. When all system parameters are determined, uEDC is a deterministic problem as follows:

$$\textbf{DP}: \quad \min \quad \mathcal{C} + \beta\mathcal{L} \qquad \text{s.t.} \quad (3) \tag{4}$$

where β serves as a variable balancing the importance of storage cost and data retrieval latency. In real-time video streaming applications, users have a high demand for data access timeliness, so a larger β can be chosen. In contrast, for backup update applications, the frequency of data access may be relatively low, allowing for a smaller β to achieve a more cost-effective data caching strategy.

Considering server failures and data request uncertainty, the solutions obtained from the deterministic model DP may fall short in real EDC scenarios, potentially resulting in excessive data retrieval latency. Robust optimization is an effective method for addressing uncertainties, which uses the *uncertainty set* to model the uncertain parameters [29]–[31]. The uEDC problem can be modeled as a single-stage robust optimization problem. However, due to the nonlinear constraints associated with server failures and the linear constraints related to data request uncertainties, applying a single-stage robust optimization approach often results in overly cautious solutions within the uncertainty set. To address this issue, we introduce a two-stage robust optimization approach. The core idea is to partition the uEDC problem into two phases: the initial stage and a subsequent stage which formulates a refined decision strategy based on the outcome of the first stage. This approach aims to mitigate the impacts of uncertainty set on data caching decisions.

In the first phase, initial data caching decisions are made based on data popularity under capacity constraints (Eq. (3)). In the second phase, we incorporate actual user request fluctuations and server failures to optimize the decisions made in the first phase based on the extent of request fluctuations and server failures. Specifically, the cases where data requests cannot be met due to fluctuations and server failures are categorized into two types: requests that can be served by other servers within the ESS under the latency constraint[1] $\mathcal{L}_{max}$ are considered as satisfiable requests $m_{i,j}$, while other requests $q_{i,j}$ are considered as unsatisfiable requests by the ESS. Consequently, by integrating these two phases, the data retrieval latency also consists of two components: the first-stage estimated latency $\mathcal{L}^1$ and the second-

1. To ensure an acceptable data retrieval latency for users, the transmission latency between edge servers must not exceed an application-specific latency constraint $\mathcal{L}_{max}$ [32], [33], which can be predefined by app vendors according to the applications' requirements.

stage actual latency $\mathcal{L}^2$. These two latency components are expressed as follows:

$$\mathcal{L}^1 = \sum_{i,k} a_{i,k} p_{j,k} \tag{5}$$

$$\mathcal{L}^2 = \sum_{i,j} q_{i,j} t_j + \sum_{i,j} m_{i,j} p_{j,k} \tag{6}$$

The real data request can be formalized as an interval, i.e., $r_i^j \in [\hat{r}_{i,j} + g_i^j \bar{r}_{i,j}]$, where $\hat{r}_{i,j}$ denotes the expected value of data request and $\bar{r}_{i,j}$ denotes the maximum deviation between real data request and the expected value. Therefore, the uncertainty set of data requests can be formulated as follows:

$$\mathcal{R} = \{r : r_{i,j} = \hat{r}_{i,j} + g_i^j \bar{r}_{i,j}; \forall d_i \in D, \forall v_j \in V, \forall g \in G\} \tag{7}$$

where $G = \{g : \sum_{i,j} g_i^j \leq P, 0 \leq g_i^j \leq 1\}$ and P is the budget of data request uncertainty. When $P = 0$, the users' data request is equal to the expected one. In this way, this robust optimization model is thus a deterministic model.

In addition to request uncertainty, app vendors must consider unforeseeable edge server failures during the decision-making process. To address the uncertainty surrounding server failures, we adopt a cardinality-constrained uncertainty set similar to [29] and [34]. Let w_j be a binary indicator, where $w_j = 0$ if edge server v_j fails. The uncertainty set of edge server failures can be defined as follows:

$$\mathcal{W} = \{w_j \in \{0,1\} : \sum_j w_j \leq F\} \tag{8}$$

where $F (F \leq |V|)$ represents the maximum number of edge servers that may fail in the ESS.

2.2 Problem Formulation

By incorporating uncertainties into the deterministic problem, we address potential disruptions, such as server failures that may prevent data requests from being processed locally, necessitating offloading to a remote cloud, as defined in Eq. (6). The two-stage robust optimization model for the uEDC problem can be formulated as follows:

$$\mathbf{RP}: \quad \min_a \{\sum_{i,k} s_k a_{i,k} + \beta \sum_{i,k} a_{i,k} p_{j,k}\}$$
$$+ \max_{r,w} \min_{m,q} \sum_{i,j} q_{i,j} t_j + \sum_{i,j} m_{i,j} p_{j,k} \tag{9a}$$
$$\text{s.t.} \quad \sum_i a_{i,k} \leq C_k \tag{9b}$$
$$\sum_i m_{i,j} \leq w_j \sum_i r_{i,j}, \quad \forall j \tag{9c}$$
$$m_{i,j} + q_{i,j} \geq r_{i,j}, \quad \forall i, \forall j \tag{9d}$$

In the first stage, uncertainties are ignored. That is, changes in data popularity and server failures are not considered while optimizing data caching cost and data retrieval latency. A preliminary data caching strategy will be produced. In the second stage, uncertainties are introduced. The objectives are re-optimized to accommodate data requests that can be directly satisfied and those that need to be offloaded to the cloud. The first stage involves the capacity constraint (9b), while the second stage involves two constraints, i.e., (9c) and (9d). Constraint (9c) ensures that, in the event of the failure of server v_i, all data requests directed to v_i are served by data cached on servers under the $\mathcal{L}_{\max}$ latency constraint without experiencing a failure. Constraint (9d) ensures that the sum of all data requests satisfied within the system and those offloaded to the remote cloud must not be less than the total actual requests. This ensures that all data requests will be processed eventually, one way or another.

2.3 Problem Hardness

By reducing the uEDC problem from the classic $\mathcal{NP}$-hard *facility location* (FL) problem [35], we can prove the $\mathcal{NP}$-hardness of the uEDC problem.

Given a set of potential facility locations $F = \{f_1, f_2, \ldots, f_m\}$ and a set of clients $C = \{c_1, c_2, \ldots, c_n\}$, the FL problem involves determining the placement of some facilities to serve a given set of clients. The optimization objective is to minimize both the opening cost of the facilities and the cost of servicing the clients from these facilities. The FL problem can be formally described as follows:

$$\min\{\sum_{i=1}^{m} y_i g_i + \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} d_{ij}\} \tag{10a}$$
$$\text{s.t.} \quad \sum_{i=1}^{m} x_{ij} = 1, \quad \forall j \tag{10b}$$
$$x_{ij} \leq y_i, \quad \forall i, \forall j \tag{10c}$$
$$y_i \in \{0,1\}, \quad \forall i \tag{10d}$$
$$x_{ij} \in \{0,1\}, \quad \forall i, \forall j \tag{10e}$$

where y_i is a binary variable that determines whether a facility at location i is open and $x_{i,j}$ is a binary variable that determines whether client j is served by facility i.

Now we reduce the uEDC problem from the FL problem: 1) relax the data request uncertainty budget to 0; 2) relax the maximum number of edge servers fail to 0. The optimization objective of uEDC can formulated as: $\min\{\sum_{i,k} s_k a_{i,k} + \beta \sum_{i,k} a_{i,k} p_{j,k}\}$, the same as Eq. (10a). Constraint (10b) ensures each client is serviced by exactly one facility. Constraint (10c) ensures each client can only be serviced by an open facility. Combining constraint (10b) and constraint (10c), we have $\sum_{i=1,\ldots,m} x_{ij} \leq m$, which is equivalent to Eq. (9b). Combining constraints (9c) and (9d) implies that it is necessary to serve all the data requests, equivalent to all clients being connected to a facility in the FL problem.

In conclusion, any solution to the $\mathcal{NP}$-hard FL problem can be reduced to the uEDC problem in polynomial time. Thus, the uEDC problem is also $\mathcal{NP}$-hard.

3 Optimal Approach

Benders' decomposition method [36] can be employed to solve the two-stage robust optimization problem by dividing the problem into two parts: the *master problem* (MP) and the *subproblem* (SP). The MP incorporates first-stage decision variables $a_{i,j}$, constraints related only to $a_{i,j}$, and Benders' cuts returned from the SP. It also includes an auxiliary variable η for evaluating the value of the second-stage objective

function. The SP encompasses second-stage decision variables m, q and uncertainty variables r, w, aiming to provide a bound on the value of the second-stage objective function. For the two-stage robust optimization problem described in Eq. (9a), the initial MP can be formulated as follows:

$$\textbf{MP:} \quad \min_{\eta,a}\{\sum_k s_k \sum_i a_{i,k} + \beta \sum_{i,k} a_{i,k} p_{j,k}\} + \eta \tag{11a}$$

$$\text{s.t.} \quad \sum_i a_{i,k} \le C_k \tag{11b}$$

$$\eta \ge 0 \tag{11c}$$

where $\eta = \max_{r,w} \min_{m,q} \sum_{i,j} q_{i,j} t_j + \sum_{i,j} m_{i,j} p_{j,k}$. According to Eq. (9a), the SP can be formulated as a bilevel linear programming problem. In this framework, the upper level's decision variables are the uncertainty variables r and w, while the lower level's decision variables are m and q. Thus, the SP can be expressed as follows:

$$\textbf{SP:} \quad \max_{r,w} \quad \min_{m,q} \sum_{i,j} q_{i,j} t_j + \sum_{i,j} m_{i,j} p_{j,k} \tag{12a}$$

$$\text{s.t.} \quad \sum_i m_{i,j} \le w_j \sum_i r_{i,j}, \quad \forall j \tag{12b}$$

$$m_{i,j} + q_{i,j} \ge r_{i,j}, \quad \forall i, \forall j \tag{12c}$$

$$q_{i,j}(r, w) \ge 0, \quad \forall i, \forall j \tag{12d}$$

$$m_{i,j}(r, w) \ge 0, \quad \forall i, \forall j \tag{12e}$$

The upper level in Eq. (12a) optimizes decision variables r and w, influenced by uncertainties, to achieve the best possible outcome under given constraints. Concurrently, the lower level focuses on optimizing m and q, which depends on the outcome produced by the upper level. This facilitates an integrated approach to handling uncertainties effectively.

Since Eq. (12) is a max-min linear programming problem that is hard to solve, we reformulate the SP into a dual problem as follows:

$$\max_{u,s}\{\sum_{i,j} s_{i,j} r_{i,j} - u_j w_j \sum_i r_{i,j}\} \tag{13a}$$

$$\text{s.t.} \quad \sum_i s_{i,j} - u_j \le p_{j,k}, \quad \forall j \tag{13b}$$

$$\sum_i s_{i,j} \le t_j, \quad \forall j \tag{13c}$$

$$u_j \ge 0, \quad \forall j \tag{13d}$$

$$s_{i,j} \ge 0, \quad \forall i, \forall j \tag{13e}$$

where $s_{i,j}$ and u_j are dual variables. Using the standard Benders dual theory [37], the max-min SP can be transformed into a single max problem.

By combining Eq. (13) and Eq. (12), we can derive a unified maximization problem as Eq. (14). Solving Eq. (12) is equivalent to solving Eq. (14).

$$\max_{r,w,u,s} \sum_{i,j} s_{i,j} r_{i,j} - u_j w_j \sum_i r_{i,j} \tag{14a}$$

$$\text{s.t.} \quad \sum_i s_{i,j} - u_j \le p_{j,k}, \quad \forall j \tag{14b}$$

$$\sum_i s_{i,j} \le t_j, \quad \forall j \tag{14c}$$

$$u_j \ge 0, \quad \forall j \tag{14d}$$

$$s_{i,j} \ge 0, \quad \forall i, \forall j \tag{14e}$$

Although the bilevel SP in Eq. (12) can be simplified into Eq. (14), according to the Benders' dual approach, Eq. (14) remains a bilinear optimization problem, which cannot be solved directly. To address this issue, we apply the *Karush-Kuhn-Tucker* (KKT) conditions to transform the bilinear optimization problem into a linear optimization problem for resolution.

First, we construct the Lagrangian function $\mathcal{L}$ for Eq. (14) as follows:

$$\begin{aligned}\mathcal{L}(s_{i,j}, r_{i,j}, u_j, w_j, \lambda_j^1, \lambda_j^2, \lambda_j^3, \lambda_{i,j}^4) = \sum_{i,j} s_{i,j} r_{i,j} - \\ u_j w_j \sum_i r_{i,j} + \lambda_j^1 (\sum_i s_{i,j} - u_j - p_{j,k}) + \\ \lambda_j^2 (\sum_i s_{i,j} - t_j) - \mu_j^1 u_j - \mu_{i,j}^2 s_{i,j} - \mu_{i,j}^3 r_{i,j} - \mu_j^4 w_j\end{aligned} \tag{15}$$

where λ and μ are the Lagrange multipliers associated with the constraints on variables q and m, respectively.

The obtained KKT conditions include *stationary conditions* as follows:

$$\frac{\partial \mathcal{L}}{\partial s_{i,j}} = r_{i,j} + \lambda_j^1 + \lambda_j^2 - \lambda_{i,j}^4 = \mu_{i,j}^2, \quad \forall i, j \tag{16a}$$

$$\frac{\partial \mathcal{L}}{\partial r_{i,j}} = s_{i,j} + u_j w_j = \mu_{i,j}^3, \quad \forall i, j \tag{16b}$$

$$\frac{\partial \mathcal{L}}{\partial u_j} = -w_j \sum_i r_{i,j} - \lambda_j^1 = \mu_j^1, \quad \forall j \tag{16c}$$

$$\frac{\partial \mathcal{L}}{\partial w_j} = -u_j \sum_i r_{i,j} = \mu_j^4, \quad \forall j \tag{16d}$$

The *primal feasibility conditions* can be formulated as follows:

$$\sum_i s_{i,j} - u_j \le p_{j,k}, \quad \forall j \tag{17a}$$

$$\sum_i s_{i,j} \le t_j, \quad \forall j \tag{17b}$$

$$u_j \ge 0, \quad \forall j \tag{17c}$$

$$s_{i,j} \ge 0, \quad \forall i, \forall j \tag{17d}$$

$$r_{i,j} \ge 0, \quad \forall i, \forall j \tag{17e}$$

$$w_j \ge 0, \quad \forall j \tag{17f}$$

The *dual feasibility conditions* are:

$$\lambda_j^1 \ge 0, \quad \forall j \quad ; \quad \lambda_j^2 \ge 0, \quad \forall j \tag{18}$$

The *complementary slackness* can be formulated as follows:

$$\lambda_j^1 (\sum_i s_{i,j} - u_j - p_{j,k}) = 0, \quad \forall j \tag{19a}$$

$$\lambda_j^2 (\sum_i s_{i,j} - t_j) = 0, \quad \forall j \tag{19b}$$

$$\mu_j^1 u_j = 0 \quad \forall j \quad \mu_{i,j}^2 s_{i,j} = 0, \quad \forall i, j \tag{19c}$$

$$\mu_{i,j}^3 r_{i,j} = 0, \quad \forall i, j \mu_j^4 w_j = 0, \quad \forall j \tag{19d}$$

After applying the KKT conditions, the problem can be expressed in the following form:

$$\max_{r,w,u,s} \sum_{i,j} s_{i,j} r_{i,j} - u_j w_j \sum_i r_{i,j} \tag{20a}$$

s.t.
$$0 \le s_{i,j} \le A_{i,j}(1 - e^1_{i,j})$$
$$0 \le r_{i,j} + \lambda^1_j + \lambda^2_j - \lambda^4_{i,j} \le A_{i,j} e^1_{i,j} \tag{20b}$$
$$0 \le r_{i,j} \le B_{i,j}(1 - e^2_{i,j}) \tag{20c}$$
$$0 \le s_{i,j} + u_j w_j \le B_{i,j} e^2_{i,j} \tag{20d}$$
$$0 \le u_j \le C_j(1 - e^1_j) \tag{20e}$$
$$0 \le -w_j \sum_i r_{i,j} - \lambda^1_j \le C_j e^1_j, \quad \forall j \tag{20f}$$
$$0 \le w_j \le D_j(1 - e^2_j) \tag{20g}$$
$$0 \le -u_j \sum_i r_{i,j} \le D_j e^2_j, \quad \forall j \tag{20h}$$
$$0 \le \lambda^1_j \le E_j(1 - e^3_j) \tag{20i}$$
$$0 \le p_{j,k} - \sum_i s_{i,j} - u_j \le E_j e^3_j, \quad \forall j \tag{20j}$$
$$0 \le \lambda^2_j \le F_j(1 - e^4_j) \tag{20k}$$
$$0 \le t_j - \sum_i s_{i,j} \le F_j e^4_j, \quad \forall j \tag{20l}$$

where $A_{i,j}, B_{i,j}, C_j, D_j, E_j, F_j$ are sufficiently-large numbers.

To solve this sub-problem, we can use an integer programming solver such as CPLEX[2] or Gurobi[3]. When the exact solution to the sub-problem is obtained, we can determine the optimal values of $\max_{r,w,u,s} \sum_{i,j} s_{i,j} r_{i,j} - u_j w_j \sum_i r_{i,j}$. With this solution, we can derive the values of r, w, q, and m corresponding to the given η. Using these values, we can regenerate a new constraint named *cutting plane*:

$$\eta \ge \sum_{i,j} q^*_{i,j} t_j + \sum_{i,j} m^*_{i,j} p_{j,k} \tag{21}$$

where $q^*_{i,j}$ and $m^*_{i,j}$ denote the solution to the SP. This cutting plane is then added to the initial MP. Each iteration yields the current model's optimal solution. As the number of cutting planes increases, the value of η increases gradually, thereby continuously improving the lower bound of the global optimal solution. The algorithm ends when the distance between the upper and lower bounds converges to less than a specified tolerance ϵ.

Algorithm 1 uEDC-B

1: Initialization: set $k = 0$, $LB = -\infty$, and $UB = +\infty$
2: **repeat**
3: Solve the SP to obtain an optimal solution $(m^{k+1,*}, q^{k+1,*}, r^{k+1,*}, w^{k+1,*})$ and update UB
4: Add a new cut into the MP and update LB
5: **until** $\frac{UB-LB}{UB} \le \epsilon$
Output: cache decisions a^*

Alg. 1 presents the pseudocode of uEDC-B. In the initial MP, there are no constraints. During the solution process of the algorithm, we continually add one of the constraints from Eq. (21) to the MP, introducing a valid cut. By solving the MP, we obtain an optimal candidate solution. The optimal solution is found by solving MP and SP interactively.

2. https://www.ibm.com/products/ilog-cplex-optimization-studio
3. https://www.gurobi.com

4 Approximation Approach

As proven in Section 2.3, the uEDC problem is $\mathcal{NP}$-hard. The uEDC-B approach can obtain a theoretically optimal solution. However, it is impractical in large-scale EDC scenarios because it would take a prohibitively long time to find the optimal solution, especially when the scale of the uncertainty set is large. To accommodate large-scale EDC scenarios, we propose an approximation approach named uEDC-L, which employs the *linear decision rule* (LDR) to narrow the search space for the uncertainty set so that an approximate solution can be found rapidly.

For RP shown in Eq. (9a), we can assume that the second-stage decision variables m and q are affine functions of r and w. Thus, the model RP can be rewritten as follows.

$$\textbf{RP'}: \min_{a,c}\{\sum_{i,k} s_k a_{i,k} + \beta \sum_{i,k} a_{i,k} p_{j,k}\} + \max_{(r,w)\in\Upsilon} \min_{m,q} \sum_{i,j} q_{i,j}(r,w) t_j + \sum_{i,j} m_{i,j}(r,w) p_{j,k} \tag{22a}$$

s.t.
$$\sum_i m_{i,j}(r,w) \le w_j \sum_i r_{i,j}, \quad \forall j \tag{22b}$$
$$m_{i,j}(r,w) + q_{i,j}(r,w) \ge r_{i,j}, \quad \forall i, \forall j \tag{22c}$$
$$q_{i,j}(r,w) \ge 0, \quad \forall i, \forall j \tag{22d}$$
$$m_{i,j}(r,w) \ge 0, \quad \forall i, \forall j \tag{22e}$$

Since m and q are only related to the uncertainty variables r and w, the linear decision rule can be defined as follows:

$$q_{i,j}(r,w) = q^0_{i,j} + \sum_{\delta\in[I_r]} q^1_{i,j} r_\delta + \sum_{\epsilon\in[I_w]} q^2_{i,j} w_\epsilon, \forall i, \forall j \tag{23}$$

$$m_{i,j}(r,w) = m^0_{i,j} + \sum_{\delta\in[I_r]} m^1_{i,j} r_\delta + \sum_{\epsilon\in[I_w]} m^2_{i,j} w_\epsilon, \forall i, \forall j \tag{24}$$

where I_r and I_w denote the uncertainty sets in Eq. (22), q^0, q^1, q^2 and m^0, m^1, m^2 denote the determinable constants.

Based on the linear decision rule, RP' shown in Eq. (22)

can be reformulated as follows:

$$\textbf{RP*}: \min_{a,c,q^{0,1,2},m^{0,1,2}} \{\sum_{i,k} s_k a_{i,k} + \beta \sum_{i,k} a_{i,k} p_{j,k} + \zeta\} \quad (25a)$$

$$\text{s.t.} \quad \zeta \geq \sum_{i,j} t_j (q^0_{i,j} + \sum_{\delta} q^1_{i,j} r_\delta + \sum_{\epsilon} q^2_{i,j} w_\epsilon) + \sum_{i,j} p_{j,k} (m^0_{i,j} + \sum_{\delta} m^1_{i,j} r_\delta + \sum_{\epsilon} m^2_{i,j} w_\epsilon) \quad (25b)$$

$$\sum_{i} (m^0_{i,j} + \sum_{\delta} m^1_{i,j} r_\delta + \sum_{\epsilon} m^2_{i,j} w_\epsilon) \leq w_j \sum_{i} r_{i,j} \quad (25c)$$

$$(m^0_{i,j} + \sum_{\delta} m^1_{i,j} r_\delta + \sum_{\epsilon} m^2_{i,j} w_\epsilon) + (q^0_{i,j} + \sum_{\delta} q^1_{i,j} r_\delta + \sum_{\epsilon} q^2_{i,j} w_\epsilon) \geq r_{i,j} \quad (25d)$$

$$q^0_{i,j} + \sum_{\delta} q^1_{i,j} r_\delta + \sum_{\epsilon} q^2_{i,j} w_\epsilon \geq 0 \quad (25e)$$

$$m^0_{i,j} + \sum_{\delta} m^1_{i,j} r_\delta + \sum_{\epsilon} m^2_{i,j} w_\epsilon \geq 0 \quad (25f)$$

Constraints (25b) - (25f) are nonlinear in RP* so that RP* cannot be solved directly. We thus introduce dual variables to transform the original constraints into linear dual constraints, the same as the approach discussed in Section 3.

Firstly, considering Eq. (25b), we introduce dual variables μ^0, ξ^0, σ^0, and φ^0, so that Eq. (25b) can be reformulated as follows:

$$\begin{aligned}
&\zeta - \sum_{i,j} t_j q^0_{i,j} - \sum_{i,j} t_j q^1_{i,j} \hat{r}_{i,j} - \sum_{i,j} p_{i,k} m^0_{i,j} \\
&- \sum_{i,j} p_{j,k} \sum_{\delta} m_{i,j} \hat{r}_{i,j} - P\mu^0 - \sum_{\delta} \xi^0_\delta - F\sigma^0 - \sum_{\epsilon} \varphi^0_\epsilon \geq 0 \\
&\mu^0 + \xi^0_\delta \geq \sum_{\delta} t_j q^1_{i,j} \bar{r}_{i,j} + \sum_{i,j} p_{i,k} \sum_{\delta} m^1_{i,j} \bar{r}_{i,j}, \forall \delta \\
&\sigma^0 + \varphi^0_\epsilon \geq \sum_{\delta} t_j q^2_{i,j} + \sum_{i,j} p_{i,k} \sum_{\epsilon} m^2_{i,j}, \forall \epsilon
\end{aligned} \quad (26)$$

Like the reformulation of Eq. (25b), we introduce different dual variables to reformulate Eq. (25c), Eq. (25d), Eq. (25e) and Eq. (25f). Consequently, we can obtain the mixed-integer linear programming formulation of problem RP* and thus an approximate solution to RP* with a MILP solver like CPLEX or Gruobi.

4.1 Theoretical Analysis

Let Z^* be the optimal objective value of the problem RP and Z^H be the sub-optimal one, it is clear that $Z^* \leq Z^H$.

$$Z^* = \min \sum_{i,k} s_k a_{i,k} + \beta \sum_{i,k} a_{i,k} p_{j,k} + \max \eta \quad (27)$$

Eq. (27) can be solved by solving the $n+1$ domestic problems. Let x be the index of x-th domestic problem. Eq. (27) can be reformulated as follows:

$$Z^* = \min_{x=1,\ldots,n+1} G^x \quad (28)$$

where G^x can be formulated as follows:

$$G^x = P\bar{r} + \min(\sum_{i,k} s_k a_{i,k} + \beta \sum_{i,k} a_{i,k} p_{j,k} + \sum_{y=1,\ldots,x} (\bar{r} - \bar{r}_x) a^y_{i,k}) \quad (29)$$

Let a^H denote the approximate solution obtained by uEDC-L to the problem RP, and Z^H denote the sub-optimal objective value, Z^{LB} denote its lower bound objective value. Z^H can be expressed as $Z^H = Z^{LB} + \alpha Z^{LB}$, where α is the gap parameter between Z^H and Z^{LB}.

For each objective function value z^H_x of the deterministic problem, it is clear that (for ease of understanding, we omit all the subscripts of the variables in the original problem):

$$Z^H \leq Z^H_x = \min \sum s a^H + \beta \sum a^H p + \max \eta(a^H) \quad (30)$$

Based on dual theory, the inner maximization problem can be replaced with:

$$\max \eta(a^H) = \min\{\sum_{x=1,\ldots,n+1} \max(\bar{r}_x - \theta, 0)(a^H) + P\theta\} \quad (31)$$

The above equation can be appropriately scaled so that when θ is set to r_x, the following can be obtained:

$$\begin{aligned}
&\min\{\sum_{x=1,\ldots,n+1} \max(\bar{r}_x - \theta, 0)(a^H) + P\theta\} \\
&\leq P\bar{r}_l + \sum_{x=1,\ldots,n+1} (\bar{r}_x - \bar{r}_l)(a^H)
\end{aligned} \quad (32)$$

Combining Eq. (28) and Eq. (30), we can derive:

$$\min \sum s a^H + \beta \sum a^H p + \max \eta(a^H) \leq G^x \quad (33)$$

Combining Eq. (32) and Eq. (33), we can derive:

$$Z^H \leq Z^H_x \leq (\alpha + 1) Z^* \quad (34)$$

Therefore, the approximation ratio for uEDC-L is $\alpha + 1$ where α is given by $\max_{x=1,\ldots,n+1} \alpha_x$.

5 Evaluation

In this section, we evaluate the performance of uEDC-B and uEDC-L experimentally on the real-world traces. We implement them in Java 17 for testing in a testbed edge storage system, focusing on their data caching cost and data retrieval latency.

5.1 Experiment Setup

Trace and System Setup. All the experiments are conducted on the Telecom Shanghai Dataset [38], which includes detailed records of over 7.2 million mobile internet access instances from 9,481 mobile devices across 3,233 base stations over six months. Inspired by [15], we randomly set the data cache storage capacity for each edge server, with a maximum limit of 100 units, and each data item is randomly sized from 1 to 4 units. The parameters F, P, and β are set to 3, 5, and 1, respectively, as the default values. Based on the traces, we cache the 1,000 most popular videos we collected in September 2023 in the edge storage system. The edge storage system consists of 30 edge servers randomly chosen in an area from the Telecom Shanghai dataset, as visualized in Fig. 3. The implementations of uEDC-B and uEDC-L run on a computer equipped with Intel i9-13900k CPU (8 performance cores) and 64G RAM. They employ the classic solver CPLEX to solve the *mixed-integer linear programming* (MILP) formulations. Each experiment is repeated 100 times and the average values are reported.

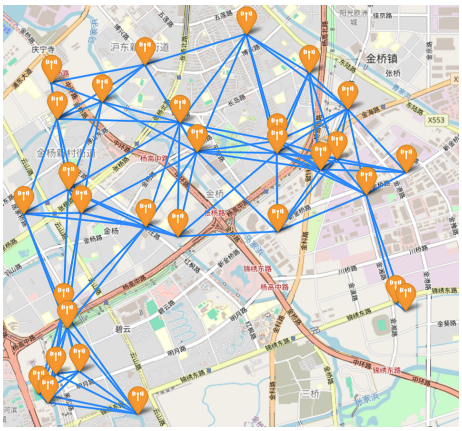


Fig. 3: Real-world edge storage system

Competing Approaches. We evaluate uEDC-B and uEDC-L against three representative EDC approaches including one baseline approach and two state-of-the-art approaches.

- **Random.** This baseline approach randomly caches data on edge servers until it does not violate the capacity constraint.
- **MagNet [39].** This deterministic EDC approach caches popular data in the system without considering data popularity uncertainty or edge server failures. MagNet first utilizes a neural embedding algorithm to capture historical content request patterns and clustering contents. Then, it caches data in the system based on predicted data popularity while balancing the storage utilization across edge servers.
- **ARO [29].** This state-of-the-art uEDC solution adopts a two-stage robust optimization method based on the classic column-and-constraint generation algorithm to achieve optimal service placement and load allocation under a given budget. In ARO, each data request can only be served by the edge server that covers it directly. We set the budget constraint in ARO as the maximum capacity of the edge servers in the system to facilitate a fair comparison.

Three versions of uEDC-L are implemented in the experiments to evaluate the role of popularity uncertainty and edge server failures in large-scale uEDC scenarios.

- **uEDC-L$_0$.** This approach does not consider any uncertainties, and thus solves the deterministic problem formulated in Eq. (4).
- **uEDC-L$_1$.** This approach removes the consideration of edge server failures. Specifically, it sets $w_j = 0$ in problem RP for all the edge servers.
- **uEDC-L$_2$.** This approach removes the consideration of data popularity uncertainty. Specifically, it assumes that the data requests equal the predicted value following a Poisson distribution, which is similar to [40] and [41].

5.2 Overall Evaluation

Table. 2 compares the overall performance of Random, MagNet, ARO, uEDC-B, and uEDC-L, measured by data caching cost calculated by Eq. (6) and data retrieval latency calculated by Eq. (2). We can see that:

TABLE 2: Overall performance of different approaches (System size = 30, $F = 3$, $P = 5$)

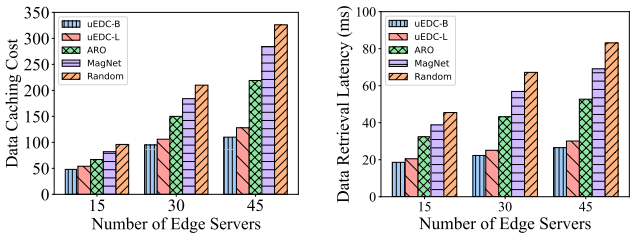
Approaches	Data caching cost	Data retrieval latency (ms)
Random	269.78	77.31
MagNet	229.11	65.37
ARO	168.56	41.78
uEDC-L	112.26	26.57
uEDC-B	**96.26**	**24.40**

- In terms of data caching cost, uEDC-B achieves the best performance among the five approaches. Compared to other approaches, it demonstrates the following cost advantages: 64.3% lower than Random, 58.0% lower than MagNet, 42.9% lower than ARO, and 14.3% lower than uEDC-L. This indicates the ability of uEDC-B to save costs. In contrast, Random, which does not utilize additional heuristic information, incurs the highest cost. The caching costs of MagNet and ARO fall between those of Random and uEDC-B, with ARO, which considers uncertainty, performing better than MagNet.
- In terms of data retrieval latency, uEDC-B also exhibits the best performance, with an average retrieval latency of only 24.40 milliseconds. Compared to other approaches, uEDC-B's latency is 68.4% lower than Random, 62.7% lower than MagNet, 46.7% lower than ARO, and 8.2% lower than uEDC-L. Random suffers from the highest data retrieval latency, while ARO reduces retrieval latency by 36.08% compared to MagNet. As discussed in Section 1, this confirms that the presence of uncertainty can significantly degrade the QoS for users.
- Overall, uEDC-B shows the best performance in both data caching cost and data retrieval latency, significantly outperforming other methods. uEDC-L seconds to uEDC-B, with an average performance gap of 11.25%, but still clearly superior to other methods. The performance improvement of the proposed approaches against two state-of-the-art approaches, i.e., MagNet and ARO, is as much as 59.27% in minimizing data retrieval latency and 55.07% in minimizing data caching cost.

To evaluate the performance of uEDC-B and uEDC-L more comprehensively in various large-scale EDC scenarios, we conduct a series of experiments by varying the number of edge servers, the number of failed edge servers, and the budget of request uncertainty respectively. When each of these parameters varies, the experiments are conducted 100 runs and the averaged values are reported.

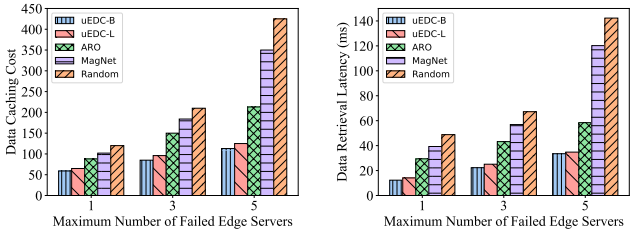
Impact of number of edge servers. As shown in Fig. 4(a), it is evident that uEDC-B consistently achieves the lowest data caching cost across all scenarios. Specifically, as the number of edge servers increases, the caching cost for all approaches rises, but uEDC-B maintains a significant cost advantage. Compared to Random, uEDC-B's cost is 64.3% lower on average. This substantial reduction is attributed to uEDC-L's efficient caching strategy that minimizes unnecessary data duplication and optimally utilizes edge servers' storage capacities. Meanwhile, as shown in Fig. 4(b), uEDC-B achieves the lowest data retrieval latency, followed closely

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2025.3531967



(a) Data caching cost (b) Data retrieval latency

Fig. 4: Performance vs. number of edge servers



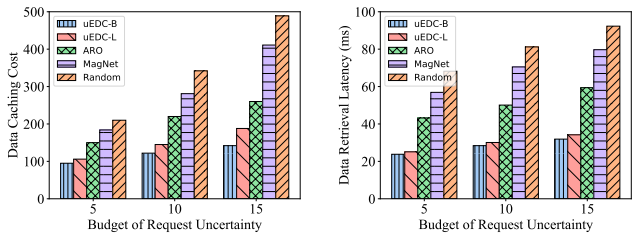
(a) Data caching cost (b) Data retrieval latency

Fig. 5: Performance vs. maximum number of failed edge servers

by uEDC-L. This indicates that the uEDC-B and uEDC-L are both highly effective in accommodating fluctuating data popularity.

Impact of number of failed edge servers. As illustrated in Fig. 2 and discussed in Section 1, edge server failures have a direct and significant impact on EDC strategies. In Fig. 5, it is evident that as the number of failed edge servers increases, the performance of all methods deteriorates in both storage cost and data retrieval latency. This degradation is attributed to users having to retrieve data from more distant neighboring servers due to edge server failures. Additionally, these strategies are compelled to cache more data in the system to mitigate the high latency associated with offloading data requests to remote clouds and thus increase data caching costs. In addition, it can be observed that the impact of the number of failed edge servers on the results is more significant than the variations caused by changes in system size as shown in Fig. 4. This underscores the critical importance of considering server failures in EDC strategies. uEDC-B and uEDC-L exhibit relatively smaller increases compared to the other three methods as the number of server failures increases, showing a linear increase. This indicates that uEDC-B and uEDC-L are highly effective in accommodating the uncertainty associated with server failures, particularly due to the collaborative caching among edge servers, which enhances data availability and reduces the impact of individual server failures.

Impact of popularity uncertainty budget. The core of EDC is to cache popular data on edge servers to serve as many user requests as possible. If data popularity uncertainty is not taken into account, it can significantly affect the effectiveness of EDC strategies. Fig. 6 illustrates the impact of increasing the request uncertainty budget on system performance. We observe that uEDC-B and uEDC-L continue to maintain a significant lead. In terms of data caching cost, uEDC-B and uEDC-L outperform ARO and MagNet by an average margin of 40.7%. For data retrieval latency, this margin increases to 55.2%. As the request uncertainty budget increases, representing greater volatility in data requests, these methods might encounter more errors, such as caching data on edge servers with fewer data requests. Consequently, the data caching cost and data retrieval latency for all methods increase. However, compared to the increases observed in other methods, uEDC-B and uEDC-L exhibit relatively smaller increments. This aligns with the results shown in Fig. 5.

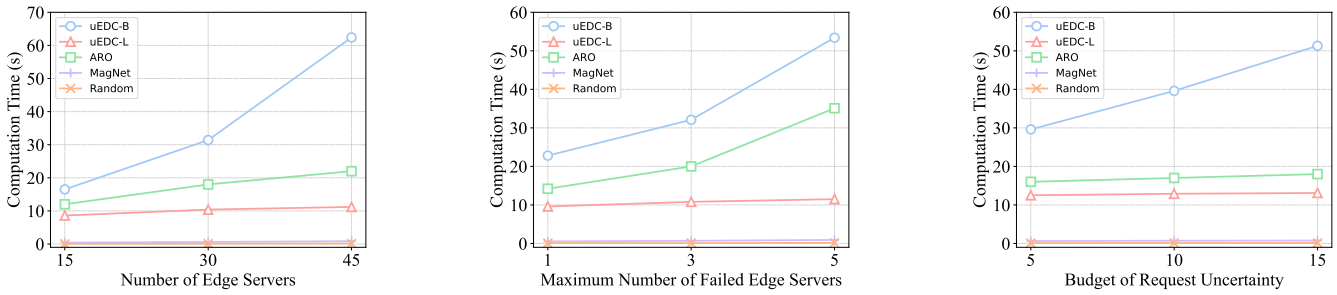


(a) Data caching cost (b) Data retrieval latency

Fig. 6: Performance vs. budget of popularity uncertainty

5.3 In-depth Evaluation

Computation Overheads. The overall analysis demonstrates the effectiveness and significant performance advantages of uEDC-B and uEDC-L in various uEDC scenarios characterized by server failures and request volatility. However, to evaluate their computation overheads associated with these performance improvements, we conducted experiments on computation time. Fig. 7 shows the results. The substantial computational overhead of uEDC-B is non-negligible. As depicted in Fig. 7(a), the computation time of uEDC-B exhibits an exponential growth trend, increasing from 16.51 seconds to 62.4 seconds as the number of edge servers rises from 15 to 45. This exponential increase is due to uEDC-B's approach of seeking the optimal solution within the entire uncertainty set, which involves extensive operations of the MILP solver, underscoring the $\mathcal{NP}$-hard nature of the uEDC problem. In contrast, uEDC-L's computational overhead is much lower. As the number of edge servers, the number of failed edge servers, and the request uncertainty budget increase, the computational overhead of uEDC-L remains relatively stable, reflecting its robustness in uncertain uEDC environments. ARO's computational overhead falls between those of uEDC-B and uEDC-L, being significantly lower than uEDC-B but on average 82.14% higher than uEDC-L. The other two methods, Random and MagNet, which do not consider uncertainty, exhibit relatively low computational overhead. As discussed in Section 4, uEDC-L achieves substantial reductions in computational overhead by employing linear decision rules, which greatly reduce the solution space of the uncertainty set, making its computational overhead only 27.86% of that of uEDC-B. Therefore, uEDC-L is a more suitable choice for solving large-scale uEDC problems than uEDC-B.

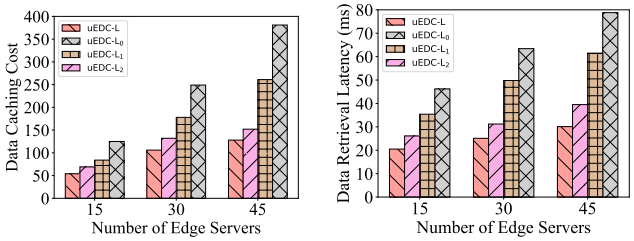


(a) Computation time vs. number of edge servers

(b) Computation time vs. number of failed edge servers

(c) Computation time vs. budget of popularity uncertainty

Fig. 7: Computation overheads comparison



(a) Data caching cost

(b) Data retrieval latency

Fig. 8: Performance comparison: with or without uncertainty modules

Impact of popularity uncertainty and server failures. Given the advantages of uEDC-L in large-scale scenarios, we further analyze the impact of popularity uncertainty and edge server failures on its performance. The differences between uEDC-L_0, uEDC-L_1, uEDC-L_2 and uEDC-L are detailed in Section 5.1. Fig. 8 illustrates that uEDC-L_2 outperforms uEDC-L_1 significantly, with an average advantage of 34.51% in terms of both data caching cost and data retrieval latency. This indicates that the impact of edge server failures on uEDC decision-making is more substantial. Predictably, the uEDC-L_0 approach, which does not consider any uncertainties, yields the poorest performance across all scenarios. Specifically, in Fig. 8(a), we observe that uEDC-L_1 incurs higher data caching costs compared to uEDC-L_2. The data retrieval latency, shown in Fig. 8(b), also follows a similar trend, with uEDC-L_1 exhibiting higher latencies across varying numbers of edge servers. The combined approach, uEDC-L, which considers both edge server failures and popularity uncertainty, is undoubtedly the most effective. It strikes a balance by efficiently handling uncertainties and failures, thus demonstrating the best performance in both data caching cost and data retrieval latency.

6 Related Work

Edge Data Caching. Edge data caching strategies have become integral to reducing latency and bandwidth usage in the mobile edge computing environment. Previous studies [17], [32], [42] focused on static caching mechanisms, assuming predictable data demand patterns. More recent studies by Wei et al. [43] and Muller et al. [44] have begun incorporating adaptive algorithms that anticipate user demands based on historical data and context-aware analysis. These methods often presume stable network conditions and consistent server availability, which are not always realistic in real-world scenarios. For example, Wei et al. [43] introduce the *similarity-aware popularity-based caching* (SAPoC) algorithm designed to enhance the performance of edge data caching in dynamic environments by leveraging content similarity. SAPoC can efficiently handle content and mobile devices' dynamic arrivals and departures, mitigating the slow-start phenomenon commonly experienced by traditional caching strategies that rely solely on request history. Machine learning techniques have been widely applied in dynamic edge data caching [45]. Sun et al. [46] proposed a decentralized, recommendation-enabled edge caching framework featuring a novel machine learning-based algorithm to minimize system costs and enhance user QoE in mobile edge-cloud networks. Wang et al. [47] introduced a cache optimization approach based on deep Q-networks, which adaptively adjusts cache strategies to maximize hit rates and reduce latency, effectively addressing the dynamic demands of edge caching. Zhang et al. [48] present a reinforcement learning-based caching strategy that utilizes the capabilities of users to assist in caching by leveraging their local caches to improve system performance and adapt to temporal and spatial variations in content popularity. By incorporating reinforcement learning, the method dynamically adjusts cache placement and content delivery policies to maximize network utility, addressing challenges posed by non-stationary content dynamics and limited caching space at small base stations. However, these studies do not account for edge server failures, and consequently, their strategies may fail to maintain performance and reliability in real-world MEC environments where server failures and network disruptions are common. This oversight can lead to increased data retrieval latency and a decrease in the overall quality of service.

Server Failure in MEC Environments. Server reliability is a critical challenge in dynamic mobile edge computing environments. Qiu et al. [49] have explored resilience strategies, focusing on maintaining service continuity in the presence of edge server failures. They combine mobile edge computing with network function virtualization to provide

dynamic *service function chains* (SFCs) using *virtual network functions* (VNFs) on resource-limited edge servers. Given the fluctuating reliability of VNFs due to workload and time variations, the paper addresses how to optimally deploy and back up VNFs to enhance SFC reliability, maximize throughput, and minimize deployment and communication costs. Fondo-Ferreiro et al. [50] highlight a gap in dynamically relocating applications across edge servers with service continuity. The proposed solution employs *software-defined networking* (SDN) to dynamically reroute users' requests to new edge infrastructure locations by creating a new IP anchor point instance. This approach includes a method using SDN to replicate previous connection contexts at the new anchor point, ensuring session and service continuity. Yet, these studies do not integrate their solutions with caching strategies, leaving a gap in holistically addressing performance and reliability in edge data caching scenarios. Another important contribution by Chang et al. [51] and Huang et al. [42] delves into redundancy techniques that replicate data across multiple edge servers. It is not difficult to see that such strategies can easily result in excessive storage costs.

Popularity Uncertainty. The assumption of predictable data popularity has been questioned in recent literature, with scholars like Xia et al. [32], Yang et al. [18] and Abolhassani et al. [52] demonstrating significant fluctuations in users' data demands that affect caching efficiency. For example, Yang et al. [18] explore location-aware edge data caching, focusing on predicting content popularity to optimize caching strategies. They develop a linear model to predict content hit rates by factoring in both content features and location characteristics, addressing the challenge of dynamic content popularity. The proposed solution incorporates two novel online algorithms: a ridge regression-based algorithm for scenarios with zero-mean noise, and an H-infinity filter-based algorithm for cases with unknown noise structures. Techniques that adapt to popularity changes in real-time are discussed in their work but lack comprehensive solutions that also consider server instability.

Holistic Approaches to Edge Data Caching. A few pioneering studies have begun to explore integrated solutions to edge data caching. Samanta et al. [53] and Cheng et al. [29] proposed frameworks that adjust resource procurement and service placement strategies based on edge server failures and data demand predictions. However, their approach only considers scenarios where user requests are directly served by available servers, overlooking the potential for edge servers to collaborate on caching to optimize data storage costs. This can increase system costs substantially. These studies motivate us to design a cost-effective edge data caching method that not only tolerates server failures but also copes with the uncertainty in data popularity. Unlike existing studies that handle these aspects in isolation, our approach provides a unified solution ensuring optimal service performance with minimal resource expenditure in dynamic edge computing environments.

7 Conclusion and Future Work

Edge data caching (EDC) enables low data retrieval latency by storing popular data on edge servers close to users. Existing EDC studies usually assume that the data popularity is immutable or can be accurately predicted, without considering edge server failures. In dynamic mobile edge computing environments, these uncertainties often render existing EDC methods ineffective. In this paper, we make the first attempt to study the edge data caching problem under data popularity uncertainty and edge server failures. We model the *uncertainty-aware edge data caching* (uEDC) problem as a two-stage robust optimization problem and prove its $\mathcal{NP}$-hardness. An optimal approach uEDC-B is proposed based on Benders' decomposition method and an approximation approach uEDC-L is proposed based on linear decision rule for large-scale EDC scenarios. Both theoretical and experimental analyses show the significant performance improvement of the proposed approaches against two state-of-the-art approaches as much as 59.27% in minimizing data retrieval latency and 55.07% in minimizing data caching cost.

References

[1] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, and Y. Yang, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 515–529, 2019.

[2] W. Xiao, Y. Hao, J. Liang, L. Hu, S. A. Alqahtani, and M. Chen, "Adaptive compression offloading and resource allocation for edge vision computing," *IEEE Transactions on Cognitive Communications and Networking*, pp. 1–1, 2024.

[3] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[4] Y. Hao, L. Hu, and M. Chen, "Joint sensing adaptation and model placement in 6g fabric computing," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 7, pp. 2013–2024, 2023.

[5] W. Xiao, X. Ling, M. Chen, J. Liang, S. A. Alqahtani, and M. Chen, "Mvpoa: A learning-based vehicle proposal offloading for cloud-edge-vehicle networks," *IEEE Internet of Things Journal*, pp. 1–1, 2024.

[6] J. Zhou, F. Chen, G. Cui, Y. Xiang, and Q. He, "FEUAGame: Fairness-aware edge user allocation for app vendors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 35, no. 8, pp. 1429–1443, 2024.

[7] R. Luo, H. Jin, Q. He, S. Wu, and X. Xia, "Enabling balanced data deduplication in mobile edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 5, pp. 1420–1431, 2023.

[8] G. Zhu, Z. Lyu, X. Jiao, P. Liu, M. Chen, J. Xu, S. Cui, and P. Zhang, "Pushing ai to wireless network edge: An overview on integrated sensing, communication, and computation towards 6G," *Science China Information Sciences*, vol. 66, no. 3, p. 130301, 2023.

[9] Y. Hao, J. Wang, D. Huo, N. Guizani, L. Hu, and M. Chen, "Digital twin-assisted urllc-enabled task offloading in mobile edge network via robust combinatorial optimization," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 10, pp. 3022–3033, 2023.

[10] R. Luo, Q. He, F. Chen, S. Wu, H. Jin, and Y. Yang, "Ripple: Enabling decentralized data deduplication at the edge," *IEEE Transactions on Parallel and Distributed Systems*, 2024.

[11] T. Ouyang, K. Zhao, X. Zhang, Z. Zhou, and X. Chen, "Dynamic edge-centric resource provisioning for online and offline services co-location," in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 2023, pp. 1–10.

[12] AWS. (2023) Amazon s3 on-demand pricing. [Online]. Available: https://aws.amazon.com/s3/pricing/

[13] ——. (2023) Aws wavelength pricing. [Online]. Available: https://aws.amazon.com/wavelength/pricing/

[14] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2016, pp. 1–9.

[15] X. Xia, F. Chen, Q. He, G. Cui, J. C. Grundy, M. Abdelrazek, X. Xu, and H. Jin, "Data, user and power allocations for caching in multi-access edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 5, pp. 1144–1155, 2021.

[16] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Joint service placement and request routing in multi-cell mobile edge computing networks," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2019, pp. 10–18.

[17] Y. Liu, Q. He, D. Zheng, X. Xia, F. Chen, and B. Zhang, "Data caching optimization in the edge computing environment," *IEEE Transactions on Services Computing*, vol. 15, no. 4, pp. 2074–2085, 2020.

[18] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang, and X. S. Shen, "Content popularity prediction towards location-aware mobile edge caching," *IEEE Transactions on Multimedia*, vol. 21, no. 4, pp. 915–929, 2018.

[19] J. Zhou, F. Chen, Q. He, X. Xia, R. Wang, and Y. Xiang, "Data caching optimization with fairness in mobile edge computing," *IEEE Transactions on Services Computing*, vol. 16, no. 3, pp. 1750–1762, 2022.

[20] N. Garg, M. Sellathurai, V. Bhatia, B. Bharath, and T. Ratnarajah, "Online content popularity prediction and learning in wireless edge caching," *IEEE Transactions on Communications*, vol. 68, no. 2, pp. 1087–1100, 2019.

[21] K. Ray and A. Banerjee, "Prioritized fault recovery strategies for multi-access edge computing using probabilistic model checking," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 1, pp. 797–812, 2022.

[22] R. Luo, H. Jin, Q. He, S. Wu, and X. Xia, "Cost-effective edge server network design in mobile edge computing environment," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 4, pp. 839–850, 2022.

[23] X. Chen, G. Xu, X. Xu, H. Jiang, Z. Tian, and T. Ma, "Multicenter hierarchical federated learning with fault-tolerance mechanisms for resilient edge computing networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[24] B. Li, Q. He, F. Chen, H. Jin, Y. Xiang, and Y. Yang, "Auditing cache data integrity in the edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1210–1223, 2020.

[25] R. Luo, Q. He, M. Xu, F. Chen, S. Wu, J. Yang, Y. Gao, and H. Jin, "Edge data deduplication under uncertainties: A robust optimization approach," *IEEE Transactions on Parallel and Distributed Systems*, 2024.

[26] M. S. Elbamby, C. Perfecto, C.-F. Liu, J. Park, S. Samarakoon, X. Chen, and M. Bennis, "Wireless edge computing with latency and reliability guarantees," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1717–1737, 2019.

[27] H. Jin, R. Luo, Q. He, S. Wu, Z. Zeng, and X. Xia, "Cost-effective data placement in edge storage systems with erasure code," *IEEE Transactions on Services Computing*, 2022.

[28] T. Ouyang, X. Chen, L. Zeng, and Z. Zhou, "Cost-aware dispersed resource probing and offloading at the edge: A user-centric online layered learning approach," *IEEE Transactions on Services Computing*, 2024.

[29] J. Cheng, D. T. Nguyen, and V. K. Bhargava, "Resilient edge service placement under demand and node failure uncertainties," *IEEE Transactions on Network and Service Management*, 2023.

[30] D. T. Nguyen, H. T. Nguyen, N. Trieu, and V. K. Bhargava, "Two-stage robust edge service placement and sizing under demand uncertainty," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1560–1574, 2021.

[31] J. Cheng, D. T. A. Nguyen, and D. T. Nguyen, "Two-stage distributionally robust edge node placement under endogenous demand uncertainty," *arXiv preprint arXiv:2401.08041*, 2024.

[32] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin, "Online collaborative data caching in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 2, pp. 281–294, 2020.

[33] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng, "Multi-hop cooperative computation offloading for industrial IoT-edge-cloud computing environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2759–2774, 2019.

[34] Y. Park, S. Lee, I. Sung, P. Nielsen, and I. Moon, "Facility location-allocation problem for emergency medical service with unmanned aerial vehicle," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 2, pp. 1465–1479, 2022.

[35] L.-Y. Wu, X.-S. Zhang, and J.-L. Zhang, "Capacitated facility location problem with general setup cost," *Computers & Operations Research*, vol. 33, no. 5, pp. 1226–1241, 2006.

[36] B. Du, H. Zhou, and R. Leus, "A two-stage robust model for a reliable p-center facility location problem," *Applied Mathematical Modelling*, vol. 77, pp. 99–114, 2020.

[37] W. C. Ng, W. Y. B. Lim, Z. Xiong, D. Niyato, H. V. Poor, X. S. Shen, and C. Miao, "Stochastic resource optimization for wireless powered hybrid coded edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 3, pp. 2022–2038, 2023.

[38] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, pp. 939–951, 2019.

[39] J. Peng, Q. Li, X. Ma, Y. Jiang, Y. Dong, C. Hu, and M. Chen, "MagNet: Cooperative edge caching by automatic content congregating," in *Proceedings of the ACM The Web Conference (WWW)*, 2022, pp. 3280–3288.

[40] Y. Cai, J. Llorca, A. M. Tulino, and A. F. Molisch, "Joint compute-caching-communication control for online data-intensive service delivery," *IEEE Transactions on Mobile Computing*, 2023.

[41] B. Abolhassani, J. Tadrous, and A. Eryilmaz, "Single vs distributed edge caching for dynamic content," *IEEE/ACM Transactions on Networking*, vol. 30, no. 2, pp. 669–682, 2021.

[42] Y. Huang, X. Song, F. Ye, Y. Yang, and X. Li, "Fair and efficient caching algorithms and strategies for peer data sharing in pervasive edge computing environments," *IEEE Transactions on Mobile Computing*, vol. 19, no. 4, pp. 852–864, 2019.

[43] X. Wei, J. Liu, Y. Wang, C. Tang, and Y. Hu, "Wireless edge caching based on content similarity in dynamic environments," *Journal of Systems Architecture*, vol. 115, p. 102000, 2021.

[44] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 2, pp. 1024–1036, 2016.

[45] Z. Chang, L. Lei, Z. Zhou, S. Mao, and T. Ristaniemi, "Learn to cache: Machine learning for network edge caching in the big data era," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 28–35, 2018.

[46] C. Sun, X. Li, J. Wen, X. Wang, Z. Han, and V. C. Leung, "Federated deep reinforcement learning for recommendation-enabled edge caching in mobile edge-cloud computing networks," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 3, pp. 690–705, 2023.

[47] F. Wang, F. Wang, J. Liu, R. Shea, and L. Sun, "Intelligent video caching at network edge: A multi-agent deep reinforcement learning approach," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2020, pp. 2499–2508.

[48] X. Zhang, G. Zheng, S. Lambotharan, M. R. Nakhai, and K.-K. Wong, "A reinforcement learning-based user-assisted caching strategy for dynamic content library in small cell networks," *IEEE Transactions on Communications*, vol. 68, no. 6, pp. 3627–3639, 2020.

[49] Y. Qiu, J. Liang, V. C. Leung, X. Wu, and X. Deng, "Online reliability-enhanced virtual network services provisioning in fault-prone mobile edge cloud," *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 7299–7313, 2022.

[50] P. Fondo-Ferreiro, F. Gil-Castiñeira, F. J. González-Castaño, and D. Candal-Ventureira, "A software-defined networking solution for transparent session and service continuity in dynamic multi-access edge computing," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1401–1414, 2020.

[51] W.-C. Chang and P.-C. Wang, "Adaptive replication for mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 11, pp. 2422–2432, 2018.

[52] B. Abolhassani, J. Tadrous, A. Eryilmaz, and E. Yeh, "Fresh caching for dynamic content," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2021, pp. 1–10.

[53] A. Samanta, F. Esposito, and T. G. Nguyen, "Fault-tolerant mechanism for edge-based iot networks with demand uncertainty," *IEEE Internet of Things Journal*, vol. 8, no. 23, pp. 16 963–16 971, 2021.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2025.3531967

Ruikun Luo received his Bachelor degree from Dalian Maritime University, China in 2018. He received his PhD degree from Huazhong University of Science and Technology (HUST), China in 2023. Now he is a Postdoc at HUST. His research interests include edge computing, parallel and distributed computing, and network storage.

Zujia Zhang is currently working toward the Bachelor's degree with the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, China. Her research interests include edge computing and algorithm optimization.

Qiang He received his first PhD degree from Swinburne University of Technology, Australia, in 2009 and his second PhD degree from Huazhong University of Science and Technology, China, in 2010. He is an Associate Professor at Swinburne. His research interests include service computing, software engineering, cloud computing and edge computing. More details about his research can be found at https://sites.google.com/site/heqiang/.

Mengxi Xu as a junior student at Huazhong University of Science and Technology, she is dedicated to researching the field of edge computing. She is passionate about exploring new technologies and innovative fields, aiming to translate theoretical knowledge into practical applications. She is confident in the future development of edge computing and is committed to contributing her efforts to the advancement of this field.

Feifei Chen received the PhD degree from the Swinburne University of Technology, Australia, in 2015. She is a senior lecturer with Deakin University. Her research interests include software engineering, edge computing, cloud computing, and green computing.

Xiaohai Dai received his Ph.D degree from the School of Computer Science and Technology at Huazhong University of Science and Technology (HUST), Wuhan, China, in 2021. He is currently a postdoctoral research fellow with the School of Computer Science and Technology at HUST. His current research interests include blockchain and distributed systems. His awards include the Outstanding Creative Award in the 2018 FISCO BCOS Blockchain Application Contest and Top Ten in FinTechathon 2019.

Song Wu received the PhD degree from Huazhong University of Science and Technology (HUST) in 2003. He is a professor of computer science at HUST in China. He currently serves as the vice dean of the School of Computer Science and Technology and the vice head of Service Computing Technology and System Lab (SCTS) and the Cluster and Grid Computing Lab (CGCL) in HUST. His current research interests include cloud resource scheduling and system virtualization. He is a member of the IEEE.

Hai Jin is a Chair Professor of computer science and engineering at Huazhong University of Science and Technology (HUST) in China. Jin received his PhD in computer engineering from HUST in 1994. In 1996, he was awarded a German Academic Exchange Service fellowship to visit the Technical University of Chemnitz in Germany. Jin worked at The University of Hong Kong between 1998 and 2000, and as a visiting scholar at the University of Southern California between 1999 and 2000. He was awarded Excellent Youth Award from the National Science Foundation of China in 2001. Jin is a Fellow of IEEE, Fellow of CCF, and a life member of the ACM. He has co-authored more than 20 books and published over 900 research papers. His research interests include computer architecture, parallel and distributed computing, big data processing, data storage, and system security.