User-Distribution and Server-Heterogeneity Aware Edge Data Distribution

Guobing Zou[1], Shengyuan Bu[1], Song Yang[1], Shengxiang Hu[2], Shengye Pang[1](✉), Yanglan Gan[3], and Bofeng Zhang[4]

[1] School of Computer Engineering and Science, Shanghai University, Shanghai, China
{gbzou, shengyuanbu, yangsong, pangsy}@shu.edu.cn
[2] Shanghai Ideal Information Industry (Group) Co., Ltd. China Telecom
husx@chinatelecom.cn
[3] School of Information and Intelligent Science, Donghua University, Shanghai, China
ylgan@dhu.edu.cn
[4] School of Computer and Information Engineering, Shanghai Polytechnic University, Shanghai, China
bfzhang@sspu.edu.cn

Abstract. Edge computing reduces data access latency by distributing popular content from cloud servers to geographically dispersed edge servers near end-users. However, achieving large-scale edge data distribution incurs substantial transmission costs and time overhead. Real-world edge deployments face non-uniform user distribution and heterogeneous server resources. Although well recognized, existing approaches adopt simplified treatments of these factors, leading to two critical limitations. First, treating the target server set as predetermined without leveraging user distribution and server heterogeneity results in resource-constrained servers becoming bottlenecks while capable servers remain underutilized. Second, adopting bandwidth-centric heuristics without prioritizing high-impact servers fails to optimize delivery, delaying availability to densely populated regions. To address these limitations, we propose User-Distribution and Server-Heterogeneity Aware Edge Data Distribution (UDSH-EDD), which decomposes the problem into two stages: target server selection and data block delivery. Specifically, we develop LR-SS, a Lagrangian relaxation-based algorithm for near-optimal target server selection, and EdgePrior, a multi-factor priority scoring mechanism to optimize data block delivery. Extensive experiments on a real-world dataset demonstrate that UDSH-EDD achieves significant improvements over state-of-the-art methods in both transmission cost and user-perceived latency.

Keywords: Edge data distribution · user distribution · server heterogeneity · target server selection · data block delivery

1 Introduction

With the rapid development of 5G communication and artificial intelligence, numerous latency-sensitive applications such as autonomous driving, augmented

reality, and industrial control require real-time data access [26]. However, conventional cloud computing centralizes resources and data in geographically distant data centers, causing significant transmission latency that creates performance bottlenecks [27]. Edge computing addresses this challenge by redistributing computational and storage resources to edge locations near end-users [35]. A key component of this paradigm is the edge caching system [19], [32], [33], a distributed infrastructure of geographically dispersed edge servers serving users within their respective coverage areas. By caching popular data within this edge caching system, application providers can dramatically reduce user-perceived latency and enable near-instantaneous access, motivating extensive research on edge caching strategy design [1], [16], [11], [14], [15].

To implement these caching strategies, application vendors must distribute data from the cloud to edge caching systems. However, this process incurs significant dissemination latency and economic costs [12], [29]. Edge Data Distribution (EDD) aims to deliver a data file F from the cloud server to a set of target edge servers in the edge caching system for caching on each server, often through entry servers as intermediaries. An effective EDD solution should balance two primary objectives: efficiency by minimizing distribution latency and economy by reducing transmission costs. Various delivery approaches have been proposed to address EDD problem.

Early non-block-based approaches deliver the entire data file as a single unit, struggling to balance efficiency and economy. DataSync [2] ensures low latency via separate cloud-to-edge (C2E) transmissions to each target server but incurs high costs, while Gossip [3] minimizes costs via one C2E transmission followed by edge-to-edge (E2E) transmissions but increases latency, and EDD-A [30] attempts to reconcile both via Steiner tree construction. More recent block-based approaches partition the data file into smaller blocks as the minimum delivery unit, where target servers reconstruct the original file upon collecting sufficient blocks, further improving both objectives. Methods such as EdgeDis [12] and EdgeHydra [7] leverage parallel data block delivery and erasure coding techniques to enhance fault tolerance.

Despite these advances in addressing the efficiency-economy trade-off, real-world edge deployments face highly non-uniform user distribution and heterogeneous server resources [35], [6]. Specifically, edge servers cover geographic areas with vastly different user population densities while exhibiting substantial heterogeneity in bandwidth, storage, and computational capacity [27], [12], [7], [34]. Although well recognized, these characteristics are often simplified in existing EDD approaches, leading to two critical limitations. First, they treat the target server set as predetermined without leveraging user distribution and server heterogeneity to guide the selection process, resulting in resource-constrained servers becoming distribution bottlenecks while more capable servers remain underutilized. Second, they adopt bandwidth-centric heuristics during data block delivery without prioritizing servers serving larger user populations or possessing superior capabilities, thus do not explicitly optimize end-to-end delivery and delay content availability for densely served regions.

To solve the above limitations, in this paper, we formulate the User-Distribution and Server-Heterogeneity Aware Edge Data Distribution (UDSH-EDD) problem, which explicitly incorporates these real-world characteristics. Unlike existing approaches that assume a static target server set, it takes user distribution and server heterogeneity into account, jointly optimizing target server selection and data block delivery strategy. We tackle the target server selection by formulating it as an $\mathcal{NP}$-hard multi-constraint optimization problem and developing LR-SS, a Lagrangian relaxation-based algorithm that effectively identifies near-optimal target server sets. To better generate the data block delivery strategy, we design EdgePrior, building upon block-based data distribution paradigm, a multi-factor priority scoring model that determines entry servers and transmission scheduling to minimize transmission cost and user-perceived latency. The main contributions are:

- We model edge data distribution in practical scenarios as the UDSH-EDD problem that explicitly incorporates user distribution and server heterogeneity, and decompose it into target server selection and data block delivery stages.
- We develop LR-SS, a Lagrangian relaxation-based algorithm for near-optimal target server selection, and building upon the selected target servers, we design EdgePrior, a multi-factor priority scoring mechanism to optimize data block delivery, where both components jointly leverage user distribution and server heterogeneity.
- Extensive experiments on a real-world dataset demonstrate that LR-SS outperforms three baselines in target server selection, EdgePrior surpasses two representative EDD approaches and one state-of-the-art EDD approach in data block delivery, and their combination achieves significant overall improvements in both transmission cost and user-perceived latency.

2 Motivating Example

Figure 1 illustrates a real-world edge caching system scenario, depicting four edge servers $\{s_0, s_1, s_2, s_3\}$ serving five users $\{u_0, u_1, u_2, u_3, u_4\}$ across different geographic regions. Each server has a coverage area indicated by dashed circles and can only serve users within its range. The system needs to distribute a data file from the cloud to selected target edge servers using erasure coding [7], where each server must receive sufficient data blocks to reconstruct the complete file.

User Distribution. Users are geographically dispersed with varying density. Server s_0 covers the most users (u_0, u_1, u_2, u_3) in the densely populated region, while s_3 covers only user u_4 in an isolated area. Servers exhibit overlapping coverage—u_2 is covered by both s_0 and s_1, u_3 by s_0, s_1, and s_2, and u_4 by both s_2 and s_3—enabling users in overlapping regions to be served by multiple servers.

Server Heterogeneity. In real-world deployments, edge servers exhibit natural heterogeneity across resource capacities due to diverse network infrastructure,

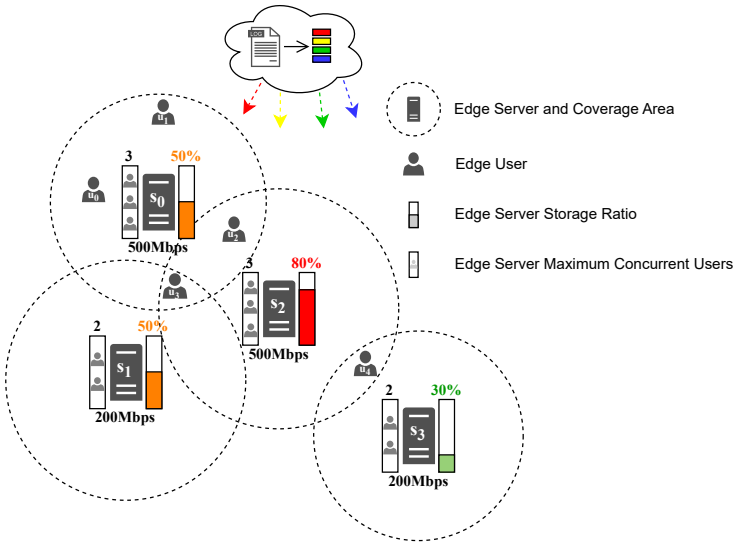


Fig. 1. A motivating example illustrating the necessity of considering user distribution and server heterogeneity in edge data distribution.

which directly impact EDD performance. In Figure 1, bandwidth varies from 200 Mbps (s_1, s_3) to 500 Mbps (s_0, s_2), computational capacity differs with maximum concurrent users ranging from 2 (s_1, s_3) to 3 (s_0, s_2), and storage availability varies substantially—s_2 is 80% occupied while others maintain 30%-50% occupancy.

Existing EDD methods typically assume target servers are pre-specified or are selected based solely on a single resource metric such as bandwidth, one might choose $\{s_0, s_2\}$ attracted by their high 500 Mbps bandwidth, and assign s_0 to serve u_0, u_1, u_2, u_3 while s_2 serves u_4. However, this selection overlooks critical constraints: s_2 has only 20% free storage risking cache thrashing, s_0 must handle four concurrent users exceeding its capacity of three, while resource-rich servers s_1 and s_3 remain idle.

Considering target servers $\{s_0, s_1, s_3\}$ are selected for data distribution, existing methods do not optimize reconstruction order, leaving completion timing arbitrary across servers. If s_3 completes reconstruction before s_0, the single user served by s_3 gains early access while multiple users in s_0's high-density region must wait longer, increasing overall user-perceived latency. Moreover, they do not exploit server heterogeneity: s_0's higher bandwidth (500 Mbps) enables faster block transmission and reconstruction than s_1 and s_3 (200 Mbps), yet receives no preferential allocation.

This demonstrates that jointly considering user distribution and server heterogeneity is essential for optimizing distribution strategies and achieving better performance.

3 System Model

3.1 UDSH-EDD Problem Definition

We consider an edge caching system consisting of a cloud server, a set of geographically distributed edge servers $S = \{s_0, s_1, \ldots, s_{n-1}\}$ with $|S| = n$, and a set of users $U = \{u_0, u_1, \ldots, u_{m-1}\}$ with $|U| = m$ scattered across different regions.

Cloud Server The cloud server cs serves as the central data repository and distribution hub [17]. It stores the original data file F with size $|F|$ (in MB). For edge caching, the cloud server encodes F using (k_e, m_e) erasure coding [7], where the original file is encoded into $n_e = k_e + m_e$ blocks: k_e data blocks and m_e parity blocks. Any k_e blocks suffice for reconstruction, providing m_e-fault tolerance. Let $B = \{b_0, b_1, \ldots, b_{n-1}\}$ denote the encoded blocks, each with size B_{size} (in MB). We assume the cloud server has sufficient computational resources for encoding and adequate bandwidth for transmissions.

User and Server Characterization S and U are geographically distributed, with positions $pos_i = (x_i, y_i) \in \mathbb{R}^2$ and $pos_j = (x_j, y_j) \in \mathbb{R}^2$, respectively. S exhibit significant heterogeneity in resource capabilities:
Definition 1 (Bandwidth heterogeneity). Available bandwidth $BW_i \in \mathbb{R}^+$ (in Mbps) for s_i determines transmission rates for both C2E and E2E data transfers, directly affecting distribution latency.
Definition 2 (Storage heterogeneity). Current storage utilization ratio $Load_i \in [0, 1]$ represents the storage occupancy of s_i, directly affecting the ability to cache new data.
Definition 3 (Computational capacity heterogeneity). Computational capabilities $Cap_i \in \mathbb{Z}^+$ for s_i measured by the maximum number of concurrent users it can serve, which results from different hardware deployment.

Coverage Relationship Each edge server has a limited coverage radius $R \in \mathbb{R}^+$ (in km), constraining its service area to nearby users. Let $dist_{ij}$ denote the Euclidean distance between user u_j and server s_i. We define a binary coverage matrix $C \in \{0, 1\}^{n \times m}$ to represent the coverage relationship:

$$C_{ij} = \begin{cases} 1, & \text{if } dist_{ij} \leq R \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where $C_{ij} = 1$ indicates that user u_j is within server s_i's coverage area and can be served by s_i.
Definition 4 (Effective distance). To model service feasibility and incorporate coverage constraints into distance measurements, we introduce the effective distance D_{ij}, assigning infinite cost to infeasible server-user pairs:

$$D_{ij} = \begin{cases} dist_{ij}, & \text{if } C_{ij} = 1 \\ +\infty, & \text{if } C_{ij} = 0 \end{cases} \tag{2}$$

Definition 5 (User-Distribution and Server-Heterogeneity Aware Edge Data Distribution Problem). A UDSH-EDD problem can be defined as a six-tuple UDSH-EDD = $\langle cs, F, U, S, G, H \rangle$, where:

- cs is the cloud server;
- F is the data file to be encoded into blocks for distribution;
- $U = \{u_0, u_1, \ldots, u_{m-1}\}$ is a set of edge users;
- $S = \{s_0, s_1, \ldots, s_{n-1}\}$ is a set of edge servers;
- $G = (C, D)$ represents the coverage relationship, where D is the effective distance between servers and users;
- $H = \{h_0, h_1, \ldots, h_{n-1}\}$ is the server heterogeneity set, where $h_i = \langle BW_i, Load_i, Cap_i \rangle$ represents bandwidth, storage, and computational capacity of server s_i;

A UDSH-EDD problem aims to distribute F from cs to edge servers S with heterogeneity H to serve users U under coverage G, minimizing transmission cost and user-perceived latency.

Since data block delivery depends on which target servers are selected, the UDSH-EDD problem is decomposes into two sequential stages.

Stage 1: Target Server Selection. This stage determines which servers to select as target servers and assigns users accordingly.

Stage 2: Data Block Delivery. Given target servers set and user assignments from Stage 1, this stage determines entry servers for C2E transmissions and transmission scheduling to minimize transmission cost and user-perceived latency.

3.2 Target Server Selection Potential Model

User Distribution Factor Users exhibit varying degrees of coverage flexibility: some are covered by multiple servers (high redundancy), while others depend on a single server (coverage-critical). We introduce a coverage competition degree to quantify each user's dependence on specific servers:

$$comp_j = \sum_{i=0}^{n-1} C_{ij} \tag{3}$$

representing the number of servers capable of serving u_j. The user distribution factor is:

$$c_{ji} = D_{ij} \cdot (comp_j)^{\delta} \tag{4}$$

where $\delta > 0$ controls the competition penalty intensity.

This formulation combines effective distance with coverage scarcity prioritization, enabling the optimization to prioritize coverage-critical users.

Server Heterogeneity Factor In edge computing, heterogeneous resources interact synergistically. To capture this complementarity among resource dimensions, we adopt the Cobb-Douglas production function [4], a classical economic model that encodes factor complementarity through multiplicative composition [21].

We quantify server s_i's selection potential as:

$$\eta_i = \left(\frac{BW_i}{BW_{max}}\right)^{\phi} \cdot \left(\frac{Cap_i}{Cap_{max}}\right)^{\psi} \cdot (1 - Load_i)^{1-\phi-\psi} \tag{5}$$

where BW_{max} and Cap_{max} represent the maximum bandwidth and computational capacity across all servers in the system. The parameters $\phi, \psi \in [0, 1]$ with $\phi + \psi \leq 1$ control the relative weighting among the three heterogeneous resource dimensions. Higher η_i indicates greater potential for selection.

The server heterogeneity factor is:

$$f_s(s_i) = \left(\frac{1}{\eta_i}\right)^{\kappa} \tag{6}$$

where $\kappa > 1$ amplifies potential differences, ensuring servers with higher η_i receive lower $f_s(s_i)$, thus prioritizing high-potential servers in the optimization.

3.3 Block Delivery Priority Model

After target servers are selected, *cs* must distribute erasure-coded data blocks [7] to these servers. We design a multi-factor priority model to optimize transmission cost and user-perceived latency.

Multi-Factor Priority Scoring We quantify each selected server's priority through a composite score integrating user assignment patterns and server heterogeneity.

User assignment factor (F_1). Measures the proportion of users served by server s_i:

$$F_1(s_i) = \frac{|U_i|}{m} \tag{7}$$

where U_i is the set of users assigned to s_i. Servers serving more users should complete reconstruction earlier to minimize aggregate user latency.

Storage availability factor (F_2). Measures available storage capacity:

$$F_2(s_i) = 1 - Load_i \tag{8}$$

Servers with higher storage availability better accommodate erasure-coded blocks [7] without triggering cache eviction overhead.

Bandwidth capacity factor (F_3). Normalizes bandwidth to $[0, 1]$ based on the range of edge server bandwidth capabilities:

$$F_3(s_i) = \max\left(0, \min\left(1, \frac{BW_i - BW_{min}}{BW_{max} - BW_{min}}\right)\right) \tag{9}$$

where BW_{min} and BW_{max} represent the minimum and maximum bandwidth across all servers. High-bandwidth servers facilitate faster data block delivery.

Combined priority score. We integrate the three factors through a weighted linear combination:

$$Score_i = \omega_1 \cdot F_1(s_i) + \omega_2 \cdot F_2(s_i) + \omega_3 \cdot F_3(s_i) \tag{10}$$

where $\omega_1, \omega_2, \omega_3 \geq 0$ are configurable weights satisfying $\omega_1+\omega_2+\omega_3 = 1$, allowing flexible prioritization based on deployment characteristics.

Transmission Cost The cloud distributes blocks to target servers through C2E and E2E transmissions. Since C2E transmissions typically incur higher costs [30], the total network transmission cost is:

$$Cost_{network} = \gamma \cdot V_{C2E} + V_{E2E} \tag{11}$$

where V_{C2E} and V_{E2E} represent the total data volumes (in MB) transmitted via C2E and E2E respectively, and $\gamma > 1$ characterizes the cost ratio between two transmission types.

User-Perceived Latency For user u_j assigned to server s_i, the end-to-end latency consists of cloud-to-server latency $T_{c2s}[s_i]$, representing the time for server s_i to complete data reconstruction by receiving at least k_e blocks, and server-to-user latency $T_{s2u}[s_i, u_j]$, modeling the last-mile delivery time from edge server to end user.

Since wireless signals propagate as electromagnetic waves, the minimum achievable delay is fundamentally constrained by the physical signal travel time over D_{ij} at the speed of light c [5]. Beyond this, practical wireless transmission incurs additional delay determined by channel capacity [9]. Following Shannon's theorem, we model the achievable channel rate Θ_{ij} as:

$$\Theta_{ij} = BW_i \cdot \log_2(1 + SNR_{ij}) \tag{12}$$

In wireless environments, signal quality degrades with distance due to path loss [5], [9]. We model this effect through a standard path loss formula where $SNR_{ij} = SNR_{ref}/(1+(D_{ij}/d_{ref})^{\beta})$, with SNR_{ref} and d_{ref} denoting reference values and β characterizing the path loss exponent. Combining propagation and transmission components, the total server-to-user latency is [7], [23]:

$$T_{s2u}[s_i, u_j] = \frac{D_{ij}}{c} + \frac{|F|}{\Theta_{ij}} \tag{13}$$

Accordingly, for user u_j assigned to server s_i, the overall user-perceived latency is:

$$Latency_j = T_{c2s}[s_i] + T_{s2u}[s_i, u_j] \tag{14}$$

4 Approach

4.1 Optimization Model for Target Server Selection

Optimization Transition Based on the quantifiable target server selection potential model established in Section 3.2, the target server selection stage aims to minimize the aggregate of user distribution factor c_{ji} for assigned user-server pairs and server heterogeneity factor $f_s(s_i)$ for selected servers. Thus, it can be transformed into an optimization problem as:

$$\min \quad \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} c_{ji} \cdot x_{ji} + \sum_{i=0}^{n-1} f_s(s_i) \cdot y_i \tag{15}$$

s.t.

$$\sum_{i=0}^{n-1} x_{ji} = 1 \tag{16}$$

$$\sum_{j=0}^{m-1} x_{ji} \leq Cap_i \cdot y_i \tag{17}$$

$$x_{ji} \leq C_{ij} \tag{18}$$

$$x_{ji} \leq y_i \tag{19}$$

where $x_{ji} \in \{0, 1\}$ and $y_i \in \{0, 1\}$ are two binary decision variables indicating whether u_j is assigned to s_i and whether s_i is selected as a target server, respectively.

The objective (15) minimizes the aggregate of user distribution factor and server heterogeneity factor. Constraint (16) ensures each user is assigned to exactly one server. Constraint (17) enforces computational capacity limits. Constraint (18) ensures users can only be served by servers that cover them. Constraint (19) ensures only target servers can serve users.

Problem Hardness Based on the formulation established above, we now prove that the target server selection is a $\mathcal{NP}$-hard problem by reduction from the Capacitated Facility Location Problem (CFLP), which is a classic $\mathcal{NP}$-hard problem [27], [18], [24].

Definition 6 (Capacitated Facility Location Problem). Given a set of facilities $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ with opening costs o_i, capacity limits cap_i, and a set of clients $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ where each client c_j has demand d_j which incurs service cost s_{ji} when served by facility f_i, the CFLP aims to select a subset of facilities to open and assign clients to them such that the total cost is minimized:

$$\min \sum_{i=1}^{n} o_i \cdot z_i + \sum_{j=1}^{m} \sum_{i=1}^{n} s_{ji} \cdot w_{ji} \tag{20}$$

s.t.

$$\sum_{i=1}^{n} w_{ji} = d_j \tag{21}$$

$$\sum_{j=1}^{m} w_{ji} \leq cap_i \cdot z_i \tag{22}$$

$$w_{ji} \leq z_i \tag{23}$$

where $z_i \in \{0, 1\}$ indicates whether facility f_i is opened and $w_{ji} \geq 0$ represents the fraction of client c_j's demand served by facility f_i. Constraint (21) ensures each client's demand is fully served. Constraint (22) enforces facility capacity limits. Constraint (23) ensures only opened facilities can serve clients.

We construct a special instance of the target server selection by mapping: (i) facilities to servers with opening costs $o_i \to f_s(s_i)$ and capacities $cap_i \to Cap_i$, (ii) clients to users with unit demands ($d_j = 1$), (iii) service costs $s_{ji} \to c_{ji}$, and (iv) setting coverage matrix $C_{ij} = 1$ for all i, j to relax coverage constraints. Under this construction, the objective (15) directly maps to (20), and constraints (16), (17), (19) correspond to (21), (22), (23) respectively. Consequently, there is a solution to the target server selection if and only if there is a solution to the corresponding CFLP. Thus, the target server selection is $\mathcal{NP}$-hard.

4.2 LR-SS Algorithm for Target Server Selection

Overview of LR-SS To solve this $\mathcal{NP}$-hard target server selection problem, we propose the LR-SS. LR-SS introduces three key innovations. First, multipliers are initialized according to user coverage competition degrees, ensuring coverage-critical users receive differentiated treatment. Second, adaptive Polyak step sizes are employed to automatically adjust optimization trajectories based on primal-dual gaps across servers with diverse capabilities. Third, greedy repair heuristics are designed to synthesize both user-side flexibility and server-side resource availability, yielding high-quality feasible solutions.

Algorithm 1 presents LR-SS's pseudo code. It initializes bounds, parameters, Lagrangian multipliers $\lambda_j^{(0)}$, user distribution factors c_{ji}, and server heterogeneity factors $f_s(s_i)$ (line 1). The main loop iterates until convergence (gap $< \epsilon$) or reaching T_{max} iterations (line 2). In each iteration, the algorithm solves a subproblem for each server s_i independently (lines 3-5): computing net benefits b_{ij} for all covered users, sorting users by descending net benefit, and greedily selecting users with positive net benefits up to capacity Cap_i, thereby determining whether to select server $y_i^{(t)}$ and which users to assign $x_{ji}^{(t)}$. The Lagrangian value $L(\lambda^{(t)})$ is evaluated to update the lower bound (line 6). When the solution violates feasibility constraints, we apply joint user-server greedy repair to resolve conflicts (lines 7-9): retaining servers with minimum c_{ji} for multi-assigned users and jointly evaluating c_{ji} and $f_s(s_i)$ when selecting servers for unassigned users, naturally exploiting both user distribution and server heterogeneity. The objective value is computed, and if it improves the current best, the upper bound is

Algorithm 1 Lagrangian Relaxation-based Server Selection (LR-SS)

Require: A UDSH-EDD problem $\langle cs, F, U, S, G, H \rangle$
Ensure: Target server selection y^* and user assignment x^*
1: Initialize $UB \leftarrow \infty$, $LB \leftarrow -\infty$, $t \leftarrow 0$, T_{max}, ϵ, $\lambda_j^{(0)}$, c_{ji}, and $f_s(s_i)$ using Eqs. (24), (4), and (6)
2: **while** $t < T_{max}$ and $(UB - LB)/UB > \epsilon$ **do**
3: **for all** servers $s_i \in S$ **do**
4: Compute net benefits and greedily select positive-benefit users up to Cap_i to determine $y_i^{(t)}$ and $x_{ji}^{(t)}$
5: **end for**
6: Evaluate $L(\lambda^{(t)})$ and update $LB \leftarrow \max(LB, L(\lambda^{(t)}))$
7: **if** the solution violates feasibility **then**
8: Repair infeasibility by retaining the minimum-c_{ji} server for multi-assigned users and assigning unserved users via minimum c_{ji} plus $f_s(s_i)$
9: **end if**
10: Compute the objective using Eq. (15), then update UB and store y^* and x^* if improved
11: Compute $g_j^{(t)} = 1 - \sum_{i=0}^{n-1} x_{ji}^{(t)}$ and update $\lambda_j^{(t+1)}$ for all users using Eq. (26)
12: $t \leftarrow t + 1$
13: **end while**
14: **return** y^* and x^*

updated and y^*, x^* are stored (line 10). Subsequently, we adopt the Polyak rule for adaptive step size adjustment, computing subgradients $g_j^{(t)}$ and updating multipliers $\lambda_j^{(t+1)}$ (line 11), which automatically adjusts step sizes based on the primal-dual gap to handle server heterogeneity. Finally, the iteration counter is updated (line 12), and the algorithm returns the best solution (y^*, x^*) (line 14).

Coverage-Based Multiplier Initialization We initialize multipliers based on coverage competition degrees to prioritize coverage-critical users:

$$\lambda_j^{(0)} = \begin{cases} \lambda_{high}, & \text{if } comp_j \leq \tau_{low} \\ \lambda_{medium}, & \text{if } \tau_{low} < comp_j \leq \tau_{high} \\ \lambda_{low}, & \text{if } comp_j > \tau_{high} \end{cases} \tag{24}$$

where $\lambda_{high} > \lambda_{medium} > \lambda_{low}$ and thresholds τ_{low}, τ_{high} categorize users by coverage scarcity. This coverage-aware initialization ensures coverage-critical users are prioritized by assigning differentiated penalties from the start, directly addressing the challenge posed by non-uniform user distribution patterns. With multipliers defined, the relaxed problem becomes:

$$\min \quad \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} (c_{ji} - \lambda_j) \cdot x_{ji} + \sum_{i=0}^{n-1} f_s(s_i) \cdot y_i + \sum_{j=0}^{m-1} \lambda_j \tag{25}$$

subject to (17), (18), (19).

Adaptive Step Size under Server Heterogeneity In each iteration, different server combinations are selected, yielding vastly different objective values caused by server heterogeneity, and thus highly dynamic primal-dual gaps. We

adopt the Polyak step size [30], [22], [8], a classical adaptive rule for subgradient optimization that automatically adjusts step sizes based on the optimality gap. This gap-based adaptation is well-suited for our problem: the gap fluctuations directly result from server heterogeneity, and the Polyak rule automatically responds to these fluctuations, enabling efficient convergence.

$$\alpha^{(t)} = \theta \cdot \frac{UB - L(\lambda^{(t)})}{\|g^{(t)}\|^2 + \epsilon} \tag{26}$$

where UB is the current best upper bound, $L(\lambda^{(t)})$ is the lower bound, $g^{(t)}$ is the subgradient vector measuring constraint violations, θ is a scaling factor, and ϵ is a small constant preventing division by zero.

Joint User-Server Greedy Repair When relaxation causes infeasibility, we employ greedy repair to construct feasible solutions. For multi-assigned users, we retain the server with minimum c_{ji}, prioritizing favorable user-server pairs based on user distribution. For unassigned users, we select servers by jointly evaluating c_{ji} and $f_s(s_i)$, simultaneously considering user coverage needs and server capabilities. This repair integrates both factors from the potential model, naturally exploiting user distribution and server heterogeneity.

4.3 EdgePrior for Priority-Based Block Delivery

Overview of EdgePrior Once target servers are selected, we propose EdgePrior, which leverages multi-factor priority scoring to determine distribution entry servers and optimize transmission scheduling, minimizing transmission cost and user-perceived latency. EdgePrior jointly considers service loads and server capabilities through priority scoring, ensuring high-priority servers reconstruct data earlier to reduce overall latency.

Algorithm 2 presents EdgePrior's pseudo code. Given target server selection y^* and user assignment x^*, the cloud server first generates $n_e = k_e + m_e$ encoded blocks using erasure coding [7] and constructs the target server set S_{target}, while initializing priority scores and allocated block counts (line 1). EdgePrior then performs two key steps to optimize data block delivery: entry server determination and transmission scheduling. For entry server determination (lines 2-3), servers are ranked by descending score, the top-$k_{percent}$ are selected as entry servers S_{entry} to receive blocks directly from the cloud, entry servers are allocated blocks proportionally to their scores, with C2E transmissions proceeding in parallel, and the maximum C2E completion time is used to initialize the transmission timeline. For transmission scheduling (lines 4-8), E2E transmissions deliver blocks to servers requiring additional blocks for reconstruction, processed sequentially by priority to ensure high-priority servers complete earlier. Finally, user-perceived latency $Latency_j$ are computed based on server completion times and user assignments (lines 9-12).

Algorithm 2 Priority-Based Block Delivery (EdgePrior)

Require: Target server selection y^*, user assignment x^*, user set U, and parameters $k_{percent}$, k_e, m_e, B_{size}, RTT_{c2e}, RTT_{e2e}, and ρ
Ensure: User-perceived latency set $\{Latency_j\}$
Generate $n_e = k_e + m_e$ encoded blocks, construct S_{target}, and initialize $Score_i$ and $|blocks_i|$ for all $s_i \in S_{target}$
Select top-$k_{percent}$ servers as S_{entry}, allocate blocks proportionally to $Score_i$, and compute $T_{c2e}^{(i)} = \frac{|blocks_i| \cdot B_{size}}{BW_i} + RTT_{c2e}$
Set $current\_time \leftarrow \max_{s_i \in S_{entry}} T_{c2e}^{(i)}$
for all servers $s_i \in S_{target}$ in priority order **do**
    $blocks\_needed \leftarrow \max(0, k_e - |blocks_i|)$
    Set $T_{c2s}[s_i] \leftarrow current\_time$ if $blocks\_needed = 0$; otherwise set $T_{c2s}[s_i] \leftarrow current\_time + \rho \left(\frac{blocks\_needed \cdot B_{size}}{BW_i} + RTT_{e2e} \right)$
    $current\_time \leftarrow T_{c2s}[s_i]$
end for
for all users $u_j \in U$ **do**
    Find the assigned server s_i with $x_{ji} = 1$ and compute $Latency_j$ by Eq. 14
end for
return $\{Latency_j\}$

Entry Server Determination EdgePrior employs the multi-factor priority scoring defined in Section 3 (Eq. 10) to rank all target servers. The top-$k_{percent}$ highest-scoring servers are designated as entry servers to receive blocks directly from the cloud:

$$S_{entry} = \{s_i \in S_{target} \mid Rank(s_i) \leq \lceil k_{percent} \cdot |S_{target}| \rceil\} \tag{27}$$

where $Rank(s_i)$ denotes rank of server s_i in descending order of $Score_i$. EdgePrior then allocates blocks to entry servers proportionally to their scores, concentrating more blocks at servers with better capabilities and higher service loads.

$$|blocks_i| = \left\lfloor \frac{Score_i}{\sum_{s_j \in S_{entry}} Score_j} \cdot n_e \right\rfloor, \quad \forall s_i \in S_{entry} \tag{28}$$

Transmission Scheduling After C2E transmissions complete, non-entry servers must acquire additional blocks through E2E transfers. EdgePrior sequences E2E transmissions according to descending priority scores, prioritizing servers serving dense user populations and possessing higher capabilities, enabling them to complete reconstruction earlier and reducing aggregate user waiting time. To capture real-world network conditions where parallel transmissions suffer from server heterogeneity and protocol overhead [27], [7], [6], we introduce serialization factor $\rho \in (0, 1]$ to model imperfect parallelism. The cloud-to-server latency is computed recursively:

$$T_{c2s}[s_r] = \begin{cases} T_{c2e}^{max}, & \text{if } |blocks_r| \geq k_e \\ T_{c2s}[s_{r-1}] + T_{transfer}^{(r)} \cdot \rho, & \text{otherwise} \end{cases} \tag{29}$$

where T_{c2e}^{max} represents the maximum C2E completion time across entry servers, $|blocks_r|$ denotes the number of blocks already allocated to server s_r, $T_{transfer}^{(r)}$

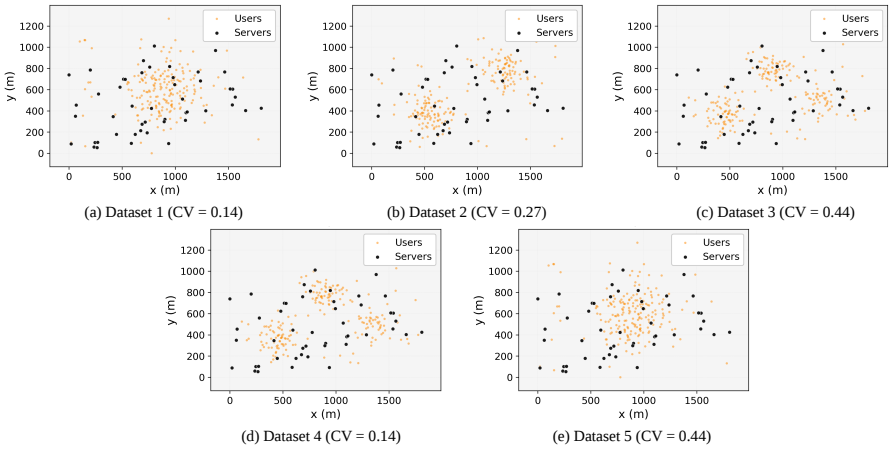


Fig. 2. Five experimental datasets under varying user distribution and server heterogeneity levels, including three aligned low/medium/high cases and two contrasting high–low and low–high cases.

denotes the E2E block transfer time for server s_r to acquire the remaining $\max(0, k_e - |blocks_r|)$ blocks needed for reconstruction, r denotes the processing order in descending priority, and $\rho \in (0, 1]$ is the serialization factor modeling the degree of parallelism: $\rho = 1$ represents fully sequential transmission, while $\rho \to 0$ approximates perfect parallelism.

5 Experiments

5.1 Experimental Setup and Datasets

All experiments were conducted on a workstation equipped with an Intel Core i7-12700H CPU running at 2.30 GHz. The components of LR-SS and EdgePrior were implemented using Python 3.12.3.

To comprehensively evaluate the performance of the combination of LR-SS and EdgePrior for edge data distribution, we conduct experiments using the EUA dataset [10][5], which comprises numerous cellular base stations in the Melbourne central business district area in Australia. We select 50 edge servers derived from the EUA dataset.

Following Gaussian distribution $\mathcal{N}(\mu, \sigma)$, 200 users are distributed in different ways. To systematically evaluate the impact of user distribution and server heterogeneity, we configure scenarios with varying levels (low, medium, and high) of both factors: (1) user distribution is characterized by different hot spot patterns, ranging from single hot spot to multiple; (2) server heterogeneity is measured

[5] `https://github.com/swinedge/eua-dataset`

through coefficient of variation (CV) of bandwidth, storage, and computational capacity, with overall CV values of approximately 0.14, 0.27, and 0.44 for three levels, respectively. The resulting five experimental datasets are illustrated in Figure 2.

5.2 Competing Methods and Evaluation Metrics

To evaluate the performance of LR-SS, we compare it against three greedy-based approaches:

- Greedy-Distance: For each user, it selects the nearest available server as the target server.
- Greedy-Bandwidth: For each user, it selects the available server with the highest bandwidth as the target server.
- Greedy-Storage: For each user, it selects the available server with the lowest load ratio as the target server.

To evaluate the performance of EdgePrior, we compare it against two representative EDD approaches and one state-of-the-art EDD approach:

- EDD-A [30]: It constructs a Steiner tree with the cloud server as the root and selected target edge servers as intermediate or leaf nodes, delivering data through tree-structured C2E and E2E transmissions to balance distribution latency and transmission costs.
- EdgeDis [12]: It partitions data into blocks and selects multiple high-bandwidth edge servers as entry points for parallel block distribution from the cloud, then introduces a coordinator to provide missing blocks and ensure all target servers can reconstruct the original data.
- EdgeHydra [7]: It encodes data into k_e data blocks and m_e parity blocks using erasure coding, distributing them to target edge servers through multiple high-bandwidth entry servers, enabling data reconstruction from any k_e available blocks even when m_e blocks are lost or delayed.

To evaluate how LR-SS and EdgePrior achieve cost-effective edge data distribution, we employ two widely used evaluation metrics:

- Transmission Cost: It quantifies the economic cost generated by network traffic during the EDD process. As defined in Equation 11, we set $\gamma = 9$ in our experiments to reflect that C2E transmissions incur significantly higher costs than E2E transmissions in practice. This metric is measured in cost per unit data.
- User-Perceived Latency: It measures the actual end-to-end latency experienced by users from the moment distribution begins to the point when data receipt is confirmed. We evaluate the average latency across all users. This metric is measured in seconds.

5.3 Experimental Results and Analysis

In the experiments, the coverage radius R is set to 1.0 km across all datasets, which guarantees that all users are within the coverage range of at least one edge server. The data file size is set to $|F| = 100$ MB, which is encoded into $k_e = 5$ data blocks and $m_e = 3$ parity blocks. For the LR-SS algorithm, there are $T_{max} = 100$, $\epsilon = 0.05$, $\lambda_{high} = 10.0$, $\lambda_{medium} = 5.0$, $\lambda_{low} = 1.0$, and $\theta = 1.5$. For the EdgePrior algorithm, there are $k_{percent} = 25\%$, $\rho = 0.6$, and $\omega_1 = 0.3$, $\omega_2 = 0.3$, $\omega_3 = 0.4$.

Table 1 presents the performance comparison results across all five datasets, covering various combinations of target server selection strategies with data block delivery approaches. For each performance metric, the optimal result is marked in bold font, while the second-best result is underlined.

Firstly, when comparing different target server selection methods while averaging across all data block delivery strategies, LR-SS consistently demonstrates superior performance. Specifically, on Dataset-1, LR-SS outperforms Greedy-Distance, Greedy-Bandwidth, and Greedy-Storage by 39.1%, 7.4%, and 3.8% in transmission cost, respectively. In terms of user-perceived latency, LR-SS is superior to Greedy-Distance, Greedy-Bandwidth, and Greedy-Storage with advantages of 47.3%, 4.0%, and 17.0%, respectively. Under the LR-SS selection strategy and comparing data block delivery approaches, EdgePrior demonstrates the most balanced performance across both metrics. In particular, EdgePrior surpasses EDD-A and EdgeHydra by 37.7% and 7.1% in transmission cost, respectively. For user-perceived latency, EdgePrior achieves improvements of 61.2%, 19.9%, and 55.6% over EDD-A, EdgeHydra, and EdgeDis, respectively. Note that EdgePrior incurs 13.3% higher transmission cost than EdgeDis due to the redundant data transmissions inherent in erasure coding for reliability guarantees, yet this overhead stays within an acceptable range considering the substantial latency reduction of 55.6%. Similar trends are observed across the remaining datasets. While single-factor greedy strategies can perform competitively in specific scenarios—for instance, Greedy-Bandwidth + EdgePrior achieves the second-best latency performance with an average gap of only 3.3% from the optimal, and Greedy-Storage + EdgeDis achieves 4.8% lower transmission cost than LR-SS + EdgeDis on Dataset-4, as moderate levels of user distribution and server heterogeneity diminish the advantages of multi-factor joint optimization, allowing the single-factor approach to remain competitive—the combination of LR-SS and EdgePrior demonstrates superior robustness by consistently securing optimal or near-optimal results across all five datasets.

Secondly, the performance of LR-SS + EdgePrior across different datasets reveals the synergistic effect of user distribution and server heterogeneity working together to influence system efficiency. Taking Dataset-1 with both factors at low levels as the baseline, we observe progressive improvements as these factors intensify jointly. In Dataset-2 with medium levels of both factors, LR-SS + EdgePrior reduces transmission cost by 4.5% and user-perceived latency by 31.6% compared to Dataset-1, with both metrics showing consistent improvements. The improvements become more pronounced in Dataset-3 where both

Table 1. Performance comparison of edge data distribution among competing methods on the EUA dataset. Bold and underlined values indicate the best and second-best results, respectively.

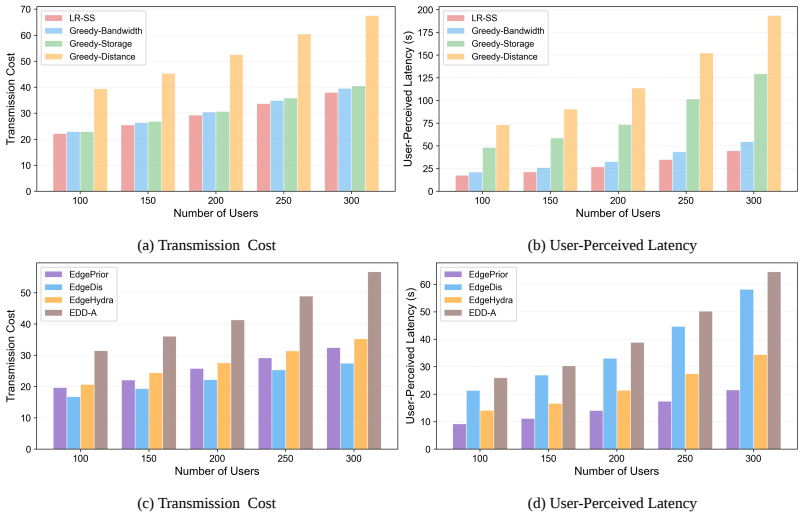
Methods	Dataset-1		Dataset-2		Dataset-3		Dataset-4		Dataset-5	
	Cost	Latency	Cost	Latency	Cost	Latency	Cost	Latency	Cost	Latency
Greedy-Distance + EDD-A	79.01	156.68	81.04	179.20	79.21	196.73	77.36	131.64	79.22	243.14
Greedy-Distance + EdgeDis	41.73	93.49	43.01	121.68	41.65	101.02	41.64	80.31	41.52	155.16
Greedy-Distance + EdgeHydra	46.93	82.51	48.02	92.89	46.64	98.31	46.73	70.00	46.39	124.38
Greedy-Distance + EdgePrior	42.54	52.09	43.44	43.08	42.81	59.36	42.77	47.91	42.89	59.61
Greedy-Bandwidth + EDD-A	48.68	74.70	43.13	47.01	43.16	47.59	46.91	72.91	45.05	46.15
Greedy-Bandwidth + EdgeDis	27.02	68.69	23.53	46.53	<u>23.40</u>	38.91	25.79	62.99	24.84	43.84
Greedy-Bandwidth + EdgeHydra	32.55	38.04	28.96	26.14	28.59	25.87	31.06	40.15	30.08	25.17
Greedy-Bandwidth + EdgePrior	29.93	<u>29.57</u>	26.88	<u>23.62</u>	26.70	<u>18.35</u>	28.90	<u>31.37</u>	27.73	<u>18.26</u>
Greedy-Storage + EDD-A	46.76	88.29	43.15	162.02	43.25	112.39	43.45	79.11	43.25	173.69
Greedy-Storage + EdgeDis	<u>25.97</u>	73.59	<u>23.42</u>	97.55	23.59	86.40	**23.44**	71.46	<u>23.24</u>	111.74
Greedy-Storage + EdgeHydra	31.31	48.12	28.81	52.28	28.91	58.80	28.69	39.80	28.73	58.47
Greedy-Storage + EdgePrior	28.99	34.20	26.87	34.55	27.11	36.46	26.97	31.74	27.13	36.12
LR-SS + EDD-A	44.96	73.75	39.53	60.74	41.36	38.87	45.04	72.92	39.70	42.04
LR-SS + EdgeDis	**24.73**	64.53	**23.14**	47.17	**22.25**	33.12	<u>24.61</u>	63.16	**22.34**	35.63
LR-SS + EdgeHydra	30.18	35.73	28.51	26.48	27.63	21.46	29.93	35.05	27.77	23.04
LR-SS + EdgePrior	28.03	**28.62**	26.77	**19.56**	25.82	**14.06**	28.00	**28.57**	26.76	**15.18**

factors reach high levels, with transmission cost decreasing by 7.9% and latency by 50.9% relative to Dataset-1, demonstrating the amplified benefits when both factors are elevated simultaneously. Note that other combinations may exhibit inconsistent performance trends across Datasets 1-3. For instance, Greedy-Storage + EdgePrior shows cost first decreasing then increasing while latency continuously worsens, and LR-SS + EDD-A displays unstable cost performance. In contrast, LR-SS + EdgePrior achieves consistent and continuous improvements in both metrics.

Finally, to isolate the individual contributions of each factor, we conduct controlled comparisons for LR-SS + EdgePrior by varying one factor while holding the other constant. Comparing Dataset-4 and Dataset-3 where user distribution remains at high level, increasing server heterogeneity from low to high reduces transmission cost by 7.8% and latency by 50.8%, indicating that server heterogeneity alone has a substantial impact on system performance. Conversely, when comparing Dataset-5 and Dataset-3 with server heterogeneity fixed at high level, elevating user distribution from low to high yields a transmission cost reduction of 3.5% and latency improvement of 7.4%, demonstrating that user distribution independently contributes to overall efficiency. Note that other combinations show limited or inconsistent responses to these factor variations. For instance, when server heterogeneity increases from Dataset-4 to Dataset-3, Greedy-Distance + EDD-A exhibits performance deterioration, while when user distribution elevates from Dataset-5 to Dataset-3, Greedy-Storage + EdgePrior shows minimal sensitivity.

5.4 Performance Impact of Parameters

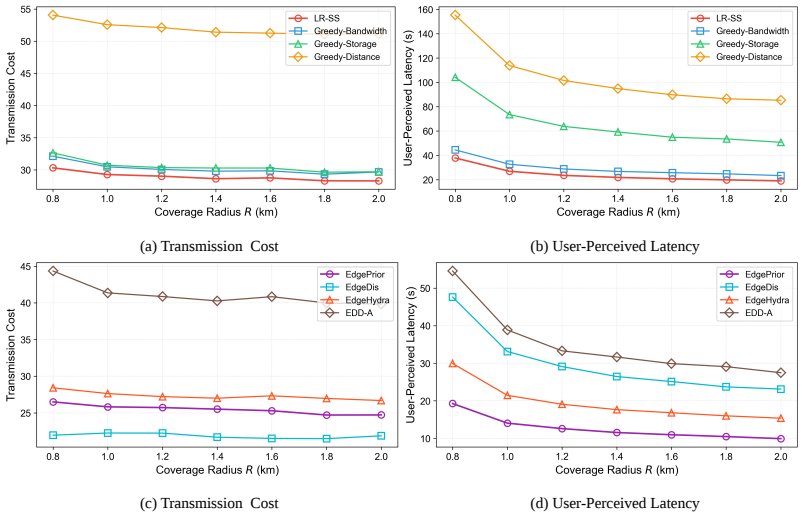
To further assess the performance of our approach for edge data distribution, we vary the following parameters on Dataset-3 and evaluate their impact on transmission cost and user-perceived latency:



(a) Transmission Cost (b) User-Perceived Latency

(c) Transmission Cost (d) User-Perceived Latency

Fig. 3. Performance impact of number of users (m).



(a) Transmission Cost (b) User-Perceived Latency

(c) Transmission Cost (d) User-Perceived Latency

Fig. 4. Performance impact of coverage radius (R).

– Number of users m. It represents the number of users in the edge cache system and is varied from 100 to 300.
– Coverage radius R. It determines the service range of each edge server and is varied from 0.8 km to 2.0 km.

The parameter experiments examine two complementary perspectives: comparing target server selection approaches by averaging across all delivery methods, and comparing data block delivery methods under fixed LR-SS selection.

Impact of Number of Users Figure 3 reveals that increasing m causes both metrics to rise as broader service areas require more target servers. As shown in Fig. 3(a), LR-SS maintains the lowest transmission cost, while single-factor baselines exhibit higher costs—Greedy-Distance selects distant servers when nearby ones reach capacity, and Greedy-Storage ignores bandwidth constraints, leading to inefficient selections. Fig. 3(b) shows sharper latency increases, with Greedy-Storage exhibiting the steepest rise as it prioritizes storage capacity alone while neglecting bandwidth and proximity, while LR-SS sustains moderate growth by balancing multiple factors. For data block delivery under fixed LR-SS, as shown in Fig. 3(c)(d), EdgePrior achieves best latency with gradual increases, while EdgeDis (no erasure coding overhead) shows lowest cost but faster latency growth. EdgePrior's priority scoring ensures heavily-loaded servers complete reconstruction earlier, reducing aggregate waiting time as user population grows.

Impact of Coverage Radius Figure 4 indicates that increasing R reduces both metrics for all methods as expanded coverage provides more server options. As shown in Fig. 4(a)(b), LR-SS consistently maintains the lowest cost and latency across all R values, while Greedy-Distance exhibits substantially higher values—larger R benefits LR-SS more significantly as it can leverage diverse server capabilities rather than being constrained by proximity-only selection. For data block delivery under fixed LR-SS, as shown in Fig. 4(c)(d), EdgePrior consistently achieves the lowest latency, while EdgeDis still shows lowest cost but higher latency. EdgePrior's adaptive scheduling maintains performance stability across varying coverage ranges by dynamically adjusting to different server pool compositions.

6 Related Work

Edge computing has shifted computational and storage resources from central clouds to the network edge, thereby transforming the design of latency-sensitive applications [28] [13]. Existing research on edge data distribution has evolved through several generations of approaches, with differing emphases [25].

Amazon [2] proposed DataSync to send data from the cloud directly to all target edge servers in parallel, minimizing latency but at high cost, as C2E transmissions typically cost five to nine times more than E2E [30][6]. Boyd et al. [3] developed Gossip protocols to minimize cost by using a single C2E entry point and relaying to other targets via E2E only, and Ongaro et al. [20] proposed Raft, where a leader receives from the cloud and forwards to others, both at the expense of increased latency from redundant E2E hops or centralized control [29].

To reconcile efficiency and economy, Xia et al. [30] proposed EDD-A by constructing Steiner trees with the cloud as root and selected edge servers as intermediate or leaf nodes, and Xia et al. [31] later formulated online cost-effective

[6] `https://aws.amazon.com/s3/pricing/`

strategies that adapt to time-varying data demand. These non-block-based approaches transmit the entire file as a single unit and remain vulnerable to runtime anomalies—data loss, network fluctuations, or node failures can compromise the whole distribution [19], [12], [7].

To enhance reliability, block-based approaches partition data into blocks as the minimum delivery unit and incorporate fault-tolerance mechanisms. Li et al. [12] proposed EdgeDis to select multiple high-bandwidth servers as entry points for parallel block distribution and use a coordinator to supply missing blocks, and He et al. [7] proposed EdgeHydra to employ erasure coding that encodes data into data blocks and parity blocks, enabling reconstruction from any sufficient subset of blocks and tolerating a fraction of lost or delayed blocks while preserving efficiency and economy.

Despite these advances in addressing the efficiency-economy-reliability trade-off, existing approaches overlook two fundamental characteristics inherent to real-world edge caching systems: user distribution and server heterogeneity. In this work, we propose a user-distribution and server-heterogeneity aware edge data distribution approach.

7 Conclusion

This paper formulated the UDSH-EDD problem, explicitly incorporating two fundamental yet underutilized characteristics of real-world edge caching systems. We decomposed UDSH-EDD into target server selection and data block delivery stages, developing LR-SS, a Lagrangian relaxation-based algorithm that effectively identifies near-optimal target server sets, and EdgePrior, a multi-factor priority scoring mechanism that determines distribution entry servers and transmission scheduling. Comprehensive experiments on a real-world dataset demonstrate that the combination of LR-SS and EdgePrior consistently achieves superior performance in transmission cost and user-perceived latency across diverse scenarios with varying user distribution and server heterogeneity levels, validating its robustness and potential applicability. In future work, we aim to extend UDSH-EDD to dynamic environments where user demand patterns and server capabilities evolve over time, enabling adaptive distribution strategies that respond to real-time system changes.

Acknowledgments. This work was supported by National Natural Science Foundation of China under Grant 62272290 and Grant 62572114.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Abolhassani, B., Tadrous, J., Eryilmaz, A., Yüksel, S.: Optimal push and pull-based edge caching for dynamic content. IEEE/ACM Transactions on Networking **32**(4), 2765–2777 (2024)

2. Amazon Web Services: AWS DataSync. `https://aws.amazon.com/datasync/` (2018), accessed: Mar. 20, 2022
3. Boyd, S., Ghosh, A., Prabhakar, B., Shah, D.: Randomized gossip algorithms. IEEE Transactions on Information Theory **52**(6), 2508–2530 (2006)
4. Cobb, C.W., Douglas, P.H.: A theory of production. The American Economic Review **18**(1), 139–165 (1928)
5. Elbamby, M.S., Perfecto, C., Liu, C.F., Park, J., Samarakoon, S., Chen, X., Bennis, M.: Wireless edge computing with latency and reliability guarantees. Proceedings of the IEEE **107**(8), 1717–1737 (2019)
6. Gao, Z., Zhang, Z., Zhang, Y., Gong, Y., Guo, Y.: Heterogeneity-aware resource allocation and topology design for hierarchical federated edge learning. IEEE Internet of Things Journal **12**(19), 39803–39816 (2025)
7. He, Q., Zhang, G., Wang, J., Luo, R., Dai, X., Hu, Y., Chen, F., Jin, H., Yang, Y.: EdgeHydra: Fault-tolerant edge data distribution based on erasure coding. IEEE Transactions on Parallel and Distributed Systems **36**(1), 29–42 (2025)
8. Kim, M., Jang, J., Choi, Y., Yang, H.J.: Distributed task offloading and resource allocation for latency minimization in mobile edge computing networks. IEEE Transactions on Mobile Computing **23**(12), 15149–15166 (2024)
9. Ko, S.W., Han, K., Huang, K.: Wireless networks for mobile edge computing: Spatial modeling and latency analysis. IEEE Transactions on Wireless Communications **17**(8), 5225–5240 (2018)
10. Lai, P., He, Q., Abdelrazek, M., Chen, F., Hosking, J., Grundy, J., Yang, Y.: Optimal edge user allocation in edge computing with variable sized vector bin packing. In: Proceedings of the 16th International Conference on Service-Oriented Computing. pp. 230–245 (2018)
11. Lekharu, A., Samanta, A., Sur, A., Patra, M.: Content-aware caching at the mobile edge network using federated learning. IEEE Transactions on Emerging Topics in Computational Intelligence **9**(2), 1166–1176 (2025)
12. Li, B., He, Q., Chen, F., Lyu, L., Bouguettaya, A., Yang, Y.: EdgeDis: Enabling fast, economical, and reliable data dissemination for mobile edge computing. IEEE Transactions on Services Computing **17**(4), 1504–1518 (2024)
13. Li, H., Sun, M., Xia, F., Xu, X., Bilal, M.: A survey of edge caching: Key issues and challenges. Tsinghua Science and Technology **29**(3), 818–842 (2024)
14. Li, H., Li, X., Sun, C., Fang, F., Fan, Q., Wang, X., Leung, V.C.M.: Intelligent content caching and user association in mobile edge computing networks for smart cities. IEEE Transactions on Network Science and Engineering **11**(1), 994–1007 (2024)
15. Li, X., Wu, Q., Fan, P., Wang, K., Cheng, N., Letaief, K.B.: Federated learning assisted edge caching scheme based on lightweight architecture DDPM. IEEE Networking Letters **7**(4), 293–297 (2025)
16. Li, Z., Shrestha, S., Song, Z., Tilevich, E.: Edge cache on WiFi access points: Millisecond-level app latency almost for free. In: IEEE International Conference on Distributed Computing Systems (ICDCS). pp. 601–612 (2024)
17. Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B.: A survey on mobile edge computing: The communication perspective. IEEE Communications Surveys & Tutorials **19**(4), 2322–2358 (2017)
18. Megiddo, N., Tamir, A.: On the complexity of locating linear facilities in the plane. Operations Research Letters **1**(5), 194–197 (1982)
19. Niu, S., Liu, Y., Liao, K., Zhang, B., Zou, G.: Dynamic edge-caching through content popularity and crowd prediction for short video services. Scientific Reports **15**, 42147 (2025)

20. Ongaro, D., Ousterhout, J.: In search of an understandable consensus algorithm. In: USENIX Annual Technical Conference (USENIX ATC). pp. 305–319. Philadelphia, PA (2014)
21. Paccagnan, D., Marden, J.R.: Utility design for distributed resource allocation-Part II: Applications to submodular, covering, and supermodular problems. IEEE Transactions on Automatic Control **67**(2), 618–632 (2022)
22. Polyak, B.T.: Minimization of unsmooth functionals. USSR Computational Mathematics and Mathematical Physics **9**(3), 14–29 (1969)
23. Sfaxi, H., Lahyani, I., Yangui, S., Torjmen, M.: Latency-aware and proactive service placement for edge computing. IEEE Transactions on Network and Service Management **21**(4), 4243–4254 (2024)
24. Shmoys, D.B., Tardos, É., Aardal, K.: Approximation algorithms for facility location problems. In: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing. pp. 265–274 (1997)
25. Tian, J., Qi, H., Li, Y., Fang, C., Qiao, J., Wang, P.: Integrated multi-dimensional prioritization and adaptive transmission for function scheduling in serverless edge computing. In: IEEE International Conference on High Performance Computing and Communications. pp. 697–704 (2024). https://doi.org/10.1109/HPCC64274.2024.00097
26. Usman, Y., Oladipupo, H., During, A.D., Akl, R., Chataut, R.: AI, ML, and LLM Integration in 5G/6G Networks: A Comprehensive Survey of Architectures, Challenges, and Future Directions. IEEE Access **13**, 168914–168950 (2025)
27. Wang, P., Jia, J., Fang, C., Zou, G., Ding, Z.: Joint data placement and service deployment in distributed cloud-edge environment. IEEE Transactions on Services Computing **18**(4), 2129–2142 (2025)
28. Wang, Y., Yang, C., Lan, S., Zhu, L., Zhang, Y.: End-edge-cloud collaborative computing for deep learning: A comprehensive survey. IEEE Communications Surveys & Tutorials **26**(4), 2647–2683 (2024)
29. Xia, Q., Xu, Z., Liang, W., Yu, S., Guo, S., Zomaya, A.Y.: Efficient data placement and replication for QoS-aware approximate query evaluation of big data analytics. IEEE Transactions on Parallel and Distributed Systems **30**(12), 2677–2691 (2019)
30. Xia, X., Chen, F., He, Q., Grundy, J., Abdelrazek, M., Jin, H.: Cost-effective app data distribution in edge computing. IEEE Transactions on Parallel and Distributed Systems **32**(1), 31–44 (2021)
31. Xia, X., Chen, F., He, Q., Grundy, J., Abdelrazek, M., Shen, J., Bouguettaya, A., Jin, H.: Formulating cost-effective data distribution strategies online for edge cache systems. IEEE Transactions on Parallel and Distributed Systems **33**(12), 4270–4281 (2022)
32. Zhao, L., Zhao, Z., Hawbani, A., Liu, Z., Tan, Z., Yu, K.: Dynamic caching dependency-aware task offloading in mobile edge computing. IEEE Transactions on Computers **74**(5), 1510–1523 (2025)
33. Zou, G., Liu, Y., Yang, S., Hu, S., Gan, Y., Zhang, B.: TEDC: Temporal-aware edge data caching with specified latency preference. In: IEEE International Conference on Web Services (ICWS). pp. 822–832 (2024)
34. Zou, G., Liu, Y., Qin, Z., Chen, J., Xu, Z., Gan, Y., Zhang, B., He, Q.: ST-EUA: Spatio-temporal edge user allocation with task decomposition. IEEE Transactions on Services Computing **16**(1), 628–641 (2023)
35. Zou, G., Yang, M., Yang, S., Pang, S., Niu, S., Gan, Y., Zhang, B.: POI-based edge service deployment with topology-aware optimization. In: IEEE International Conference on Web Services (ICWS). pp. 720–731 (2025)