

CODEA: Online Container Deployment and Edge Association for Multi-Edge Federated Synergy Learning

Shucun Fu¹, Fang Dong², *Member, IEEE*, Xiaolong Xu³, *Senior Member, IEEE*, Dian Shen⁴,
Muhammad Bilal⁵, *Senior Member, IEEE*, and Schahram Dustdar⁶, *Fellow, IEEE*

Abstract—Federated edge learning (FEEL) is a prospective paradigm enabling edge devices to collaboratively participate in machine learning model training, unlocking countless opportunities for edge intelligence. As an extension of FEEL, federated synergy learning (FSyL) alleviates the computation and communication burdens on resource-constrained end-devices by offloading partial model layers to edge servers for synergistic training. However, existing work largely ignores the impact of budgeted multi-edge service deployment and dynamic device-server association on training performance, causing significant accuracy degradation and increased costs. To address these limitations, this paper investigates critical performance bottlenecks of executing FSyL and formulates a novel benefit maximization problem that jointly optimizes online container deployment and edge association. To efficiently tackle this intractable problem, we propose *CODEA*, an online Container Deployment and Edge Association framework based on the contextual multi-armed bandit model. *CODEA* guides deploying containerized FSyL services across multiple edge servers under a budget constraint, maximizing device coverage and training robustness before each global update stage. Following container deployment, the online edge association determines the device-server association during each local training round, maximizing cost-savings and ensuring the success rate of update

collection. Extensive experiments demonstrate that *CODEA* significantly improves training accuracy and cost-savings compared to state-of-the-art methods.

Index Terms—Federated edge learning, federated synergy learning, online learning, container deployment, edge association.

I. INTRODUCTION

SIGNIFICANT advancements in distributed machine learning (DML) and mobile edge computing (MEC) have fueled the emergence of edge intelligence (EI) as a collaborative computing paradigm. EI leverages pervasive edge devices, such as IoT sensors, smartphones, and autonomous vehicles, to facilitate model training and task inference [1], [2], [3]. The widespread implementation of EI applications, including autonomous driving [4] and healthcare monitoring [5], is a crucial enabler for ubiquitous intelligent connectivity. Federated edge learning (FEEL) stands out as one of the most promising solutions for realizing EI. In FEEL, individual devices are assumed to have sufficient resources to train a complete ML model [6], [7], [8], [9]. During each training round, participating devices independently train local models on their private datasets, and the updated local models, rather than raw data, are then aggregated by a central parameter server for the next round. This iterative process is repeated across multiple rounds until a high-accuracy global model is achieved.

Existing FEEL schemes have demonstrated empirical success, providing theoretical convergence guarantees and supporting multi-level architecture designs. These advances have significantly enhanced system scalability and efficiency by expanding device coverage, alleviating network congestion, and achieving load balancing [10], [11], [12], [13]. As intelligent service demands continue to diversify, a common strategy involves augmenting deep neural network (DNN) models with increased parameters to boost the quality of service (QoS) for EI applications [14]. However, training such large-capacity DNN models directly on resource-constrained edge devices remains impractical due to the substantial mismatch between limited hardware capabilities and escalating training demands, which presents significant challenges in model deployment, resource management, and training optimization.¹

¹For instance, training ResNet18 with a modest batch size of 8 can easily exceed Android's default per-application memory limit of 0.5 GB [15].

Received 18 November 2025; revised 24 March 2026; accepted 7 April 2026. Date of publication 16 April 2026; date of current version 2 September 2026. This work was supported in part by China Postdoctoral Science Foundation under Grant 2025M781474, in part by Jiangsu Provincial Frontier Technology Research and Development Program under Grant BF2024070, in part by National Natural Science Foundation of China under Grant 62232004, Grant 62522205, and Grant 62372242, in part by Shenzhen Science and Technology Program under Grant KJZD20240903100814018, in part by Jiangsu Provincial Major Project on Basic Research of Cutting-Edge and Leading Technologies under Grant BK20232032, in part by Natural Science Foundation of Jiangsu Province under Grant BK20253020, in part by Jiangsu Provincial Key Laboratory of Network and Information Security under Grant BM2003201, in part by Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under Grant 93K-9, and in part by Collaborative Innovation Center of Novel Software Technology and Industrialization. Recommended for acceptance by M. Amadeo. (Corresponding author: Fang Dong.)

Shucun Fu and Xiaolong Xu are with the School of Computer Science, School of Software, Nanjing University of Information Science and Technology, Nanjing 210044, China (e-mail: shucunfu@nuist.edu.cn; xlxu@nuist.edu.cn).

Fang Dong and Dian Shen are with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China (e-mail: fdong@seu.edu.cn; dshen@seu.edu.cn).

Muhammad Bilal is with the School of Computing and Communications, Lancaster University, LA1 4WA Lancaster, U.K. (e-mail: m.bilal@ieee.org).

Schahram Dustdar is with Distributed Systems Group, TU Wien, 1040 Wien, Austria, and also with the ICREA, 08010 Barcelona, Spain (e-mail: dustdar@ds.g.tuwien.ac.at).

This article has supplementary downloadable material available at <https://doi.org/10.1109/TMC.2026.3684059>, provided by the authors.

Digital Object Identifier 10.1109/TMC.2026.3684059

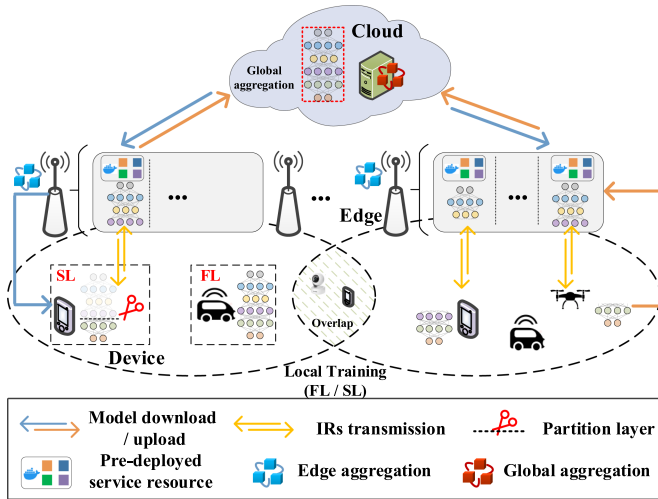


Fig. 1. Illustration of the proposed FSyL framework.

To alleviate the computation/communication burden on the resource-constrained end devices, federated synergy learning (FSyL) has emerged by integrating the strengths of split federated learning (SFL) and end-edge collaboration (E2C) [16], [17], [18]. In FSyL, the *federation*² pre-deploys training services and necessary supporting components on selected edge servers (a.k.a. container deployment). These servers function as both the parameter server for model aggregation/broadcast and the assistant server for synergistic training. Fig. 1 illustrates an example of FSyL in multi-edge scenarios, where each server is associated with nearby devices in its coverage area (a.k.a. edge association). The entire DNN model is partitioned into a device-side and a server-side sub-model at the designated cut layer. The device-side sub-model is trained locally, while the computationally intensive server-side sub-model is offloaded to the edge server for collaborative training. Despite the significant advantages of the FSyL framework for large-scale model training, two unique features (i.e., *budgeted multi-edge service deployment* and *dynamic device-server association*) of *multi-edge multi-device mapping* present significant challenges in realizing the full potential of FSyL.

Specifically, the federation needs to specify a container deployment strategy across multiple edge servers before each global update, and determine the device-server association decision for participating devices during each local training round. There are two fundamental challenges to overcome.

1) *How to determine budget-aware container deployment to ensure device coverage and training robustness?* Container deployment in multi-edge FSyL aims to maximize device coverage and increase the flexibility of subsequent device-server association, thereby enhancing training robustness. For one thing, FSyL's training performance heavily depends on the number of successfully updated devices, i.e., valid participants (see Section II-B3). While prior works have explored FL aggregator deployment to maximize device coverage [11], [12],

[19], they overlook the impact of FSyL-specific factors (i.e., sub-model offloading and inter-round strategy adaptation) on update success rates, rendering them unsuitable for dynamic FSyL scenarios. Even state-of-the-art methods [20], [21], [22], [23], [24] primarily concentrate on model partitioning and resource allocation under the assumption that FSyL services are pre-deployed across all servers. While provisioning sufficient service resources at every server could deliver the best QoS, it is impractical due to the prohibitive deployment costs and potential resource waste. For another, strategically placing FSyL services on neighboring edge servers with overlapping coverage could enhance training robustness, as devices within an overlapping coverage can dynamically switch to alternative servers when one of them overloads (see Section II-B1). Therefore, developing a budget-aware container deployment scheme that jointly considers device coverage maximization and training robustness enhancement is crucial for efficient FSyL.

2) *How to determine the device-server edge association to achieve cost-effective training?* Device-server association directly affects each device's training latency and determines whether its update can be completed and uploaded within the deadline. Due to the multi-dimensional heterogeneity of devices, the resulting sub-model workloads vary across edge servers, leading to divergent demands for edge service resources [18]. Given limited server coverage and constrained resources, device-server association decisions become interdependent and jointly impact the training time costs of participating devices (see Section II-B2). However, identifying the optimal solution is highly challenging since the combinatorial nature of device-server mappings leads to an exponentially large search space, making brute-force approaches impractical. Moreover, dynamic device participation (triggered by container deployment in each global update phase) and volatile system conditions further complicate the design of a cost-effective association strategy. Traditional numerical optimization methods struggle to handle such online decision-making problems. Therefore, once FSyL services are deployed on edge servers, it is essential to accurately estimate device training demands in a cold-start setting (i.e., without prior knowledge) and continually optimize and adapt device-server association to ensure cost-effective training and timely model updates.

Inter-server container deployment and device-server edge association are crucial for efficient training in multi-edge FSyL systems. To overcome these challenges, this paper first conducts an in-depth performance profiling analysis, examining how training accuracy and training cost are impacted under multi-edge FSyL settings. Building on these insights, we formulate an online optimization problem for FSyL container Deployment and edge Association Benefit Maximization (FDAB-M). To solve this problem, we introduce *CODEA*, an online framework for COnainer Deployment and Edge Association. *CODEA* leverages online learning techniques to make step-wise decisions during each training round, providing joint solutions that ensure efficient FSyL execution. The main contributions of this paper are summarized as follows:

- We conduct the first in-depth analysis of motivating cases to profile the training performance of multi-edge FSyL.

²Federations can be diverse entities (such as companies, research institutions, etc.) that own global ML models and initiate FL tasks.

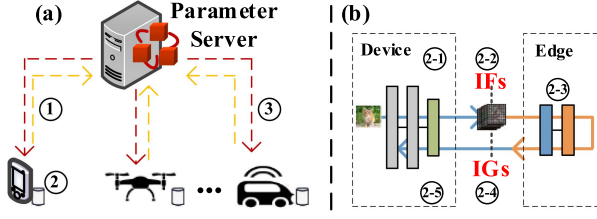


Fig. 2. Illustration of federated learning and split learning.

Based on these insights, we formulate an online optimization problem that jointly optimizes container deployment and edge association, aiming to achieve efficient and cost-effective training under multi-edge FSyL settings.

- We propose an online optimization framework, *CODEA*, built upon the contextual multi-armed bandit paradigm to decompose the long-term optimization problem into a multi-stage decision-making process. Specifically, *CODEA* initially deploys FSyL services across multiple servers under budget constraints, balancing device coverage and training robustness before each slow-timescale global update phase. Following container deployment, the edge association module is invoked during each fast-timescale local round to dynamically determine device-server association decisions, aiming to maximize cost-savings and ensure a high success rate for local update collection. By making sequential decisions throughout the training process, *CODEA* optimizes the long-term FSyL performance, providing theoretical guarantees via a sub-linear regret bound, without prior knowledge.
- We implement extensive experimental evaluations and comparative analyses on a real-world dataset within a scalable FSyL system. The results show that *CODEA* significantly outperforms state-of-the-art methods in terms of training efficiency and optimization effectiveness.

II. BACKGROUND AND MOTIVATION

A. Federated Synergy Learning Preliminary

Following the pilot effort of CHEESE [25], we adopt a general system architecture to support multi-edge FSyL. The training procedure of FSyL can be summarized as follows:

Step 1: At the beginning of each training round t , each server n distributes its edge model to the set of devices $\mathcal{S}_n$ within its coverage area. Each associated device $m \in \mathcal{S}_n$ downloads the edge model w_n^t and initializes its local model as $w_m^t \leftarrow w_n^t$.

Step 2: Then, each device m performs local training to update its model w_m^t using stochastic gradient descent (SGD) over its local dataset $\mathcal{D}_m$ as $w_m^t = w_m^t - \eta \nabla f_m(w_m^t, \mathcal{D}_m)$, where η is the learning rate, and ∇ denotes the stochastic gradient of $f_m(w_m^t)$. FSyL inherently involves device-side sub-model updates, intermediate results (IRs) communication, and server-side sub-model updates. As depicted in Fig. 2(b), the device-side sub-model belongs to the device $w_{m,d}^t$, and the server-side sub-model $w_{m,s}^t$ is hosted on the server. Specifically, the device feeds local data into its sub-model and propagates forward to the split layer (2-1). Then, the split layer's intermediate features

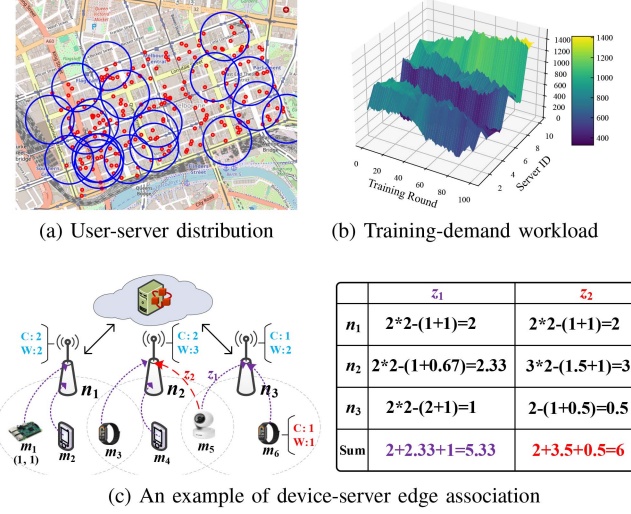


Fig. 3. Illustrations of the user-server distribution and the training demands difference.

(IFs), also called smashed data, are transmitted to the server (2-2). Following (2-2), the server receives the IFs and completes forward and backward propagation on the server-side model (2-3). Subsequently, the split layer's intermediate gradients (IGs) are sent back to the device (2-4), enabling the device to complete backpropagation and update its device-side model parameters (2-5). The device and the server collaboratively train the entire model.

Step 3: After local training, device $m \in \mathcal{S}_n$ uploads its local model w_m^t to the associated server n to be edge model w_n^t , called edge aggregation, i.e., $w_n^{t+1} = \frac{1}{|\mathcal{S}_n^t|} \sum_{m \in \mathcal{S}_n^t} w_m^t$.

Step 4: Upon reaching the global update stage (i.e., $T_g|t$), the edge model of each server will be uploaded to the cloud server for global update, which is given as $w_g^{t+1} = \frac{1}{|\mathcal{N}|} \sum_{n \in \mathcal{N}} w_n^t$.

The globally updated model is then distributed back to each edge server to initiate the next round of edge training. This process is iteratively repeated over multiple rounds until global convergence is achieved.

B. Challenges of Container Deployment and Edge Association

To analyze the factors influencing FSyL performance, we conducted a series of analyses based on real-world user-server distributions as well as simulated scenarios of container deployment and edge association, as shown in Fig. 3. For instance, Fig. 3(a) illustrates the geographic distribution of user-base stations in Metropolitan Melbourne [26]. For ease of analysis, we extracted a subset consisting of 12 base stations and 200 users from the original dataset. From the calculation results of inter-server training-demands per round (i.e., the total training time incurred by devices within the server's coverage) in Fig. 3(b), we observe significant variations in training-demands across different servers, with extreme cases reaching a $4.5\times$ difference. Moreover, service demands could fluctuate since the devices' physical location and computing/communication capacities may change during the long-term training process. These complex and dynamic factors collectively influence the effectiveness of

different container deployment and edge association strategies, with the analysis presented from three perspectives.

1) *Impact of container deployment on device coverage and training robustness in FSyL.* To investigate the impact of container deployment decisions on device coverage and training robustness, we analyzed a simulated example consisting of three edge servers and six devices, as shown in the left half of Fig. 3(c). Inter-round container deployment determines which servers are pre-configured with FSyL services under a fixed deployment budget. As shown in Fig. 3(c), we assume the available budget allows deploying FSyL services on only two servers. We compare three candidate deployment solutions, each presenting a trade-off between device coverage and training robustness. A possible solution $\mathcal{B}_1$ is to deploy FSyL services on servers n_1 and n_2 , covering five devices, all of which are expected to participate in training during the current round. Notably, device m_3 can be associated with either of the two servers, providing greater flexibility in coping with server overload. Conversely, suppose we choose the sub-optimal $\mathcal{B}_2$, i.e., deploy services on servers n_1 and n_3 . While device coverage remains at five, each device can only associate with a single server, reducing flexibility and system robustness in the event of server overload or failure. The third solution $\mathcal{B}_3$ deploys FSyL services on n_2 and n_3 . Although this option reduces device coverage to four, device m_5 gains access to two alternative servers, again improving the system's ability to adapt to overloaded servers.

Observation 1: The above analysis reveals that different container deployment strategies result in varying levels of device coverage and server co-coverage, which jointly determine the number of devices expected to participate in training and the system's robustness. Moreover, as discussed in Section II-C3, only devices that successfully complete local training and upload updates contribute positively to training accuracy. These findings motivate us to design an effective container deployment that jointly maximizes both device coverage and server co-coverage to support reliable multi-edge FSyL.

2) *Impact of device-server association on training cost in FSyL.* Device-server association determines which device is assigned to which edge server for E2C training. To illustrate this problem, we present a simplified example using unitless resource assumptions. As shown in Fig. 3(c), we assume that the training demand (including computation and communication load) of the server-side submodel requested by each device is ($c = 1, w = 1$), and each device has one unit of both computing and communication capabilities. The edge servers have heterogeneous coverage and service resources: $n_1 = (2, 2)$, $n_2 = (2, 3)$, and $n_3 = (1, 2)$, each serving a specific geographical area. We compare two edge association solutions. Considering proximity constraints, a possible association solution $\mathcal{Z}_1$ allocates m_1 and m_2 to n_1 , m_3 and m_4 to n_2 , m_5 and m_6 to n_3 . The association benefit of server n_1 is $2(\frac{1}{1} + \frac{1}{1}) - (\frac{2}{2} + \frac{2}{2}) = 2$. Similarly, the association benefits of servers n_2 and n_3 are 2.33 and 1, respectively, yielding a total benefit of 5.33. Conversely, if we choose the better $\mathcal{Z}_2$ (i.e., switch m_5 from n_3 to n_2), then the total association benefit becomes $2 + 3.5 + 0.5 = 6$. This improvement occurs since n_2 is more powerful than n_1 and n_3 , and devices m_3 and m_5 prefer to connect to n_2 to minimize

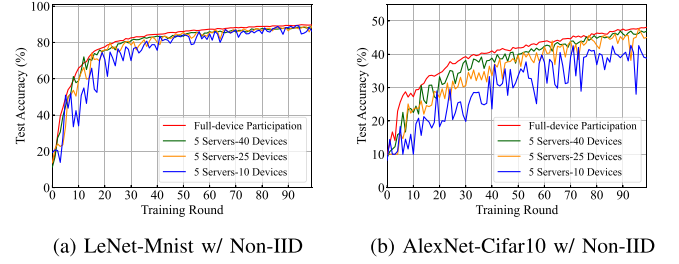


Fig. 4. Training accuracy of different device participation.

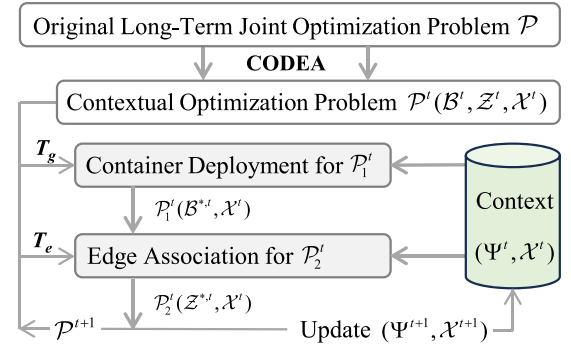


Fig. 5. The design of CODEA.

training costs. Such an edge association problem for sub-model offloading can be modeled as a *constrained multi-dimensional knapsack problem*. In practice, numerous feasible solutions exist, and finding the one that maximizes overall association benefit is non-trivial. The problem is further complicated by dynamic device locations and varying training demands, which directly affect system cost.

Observation 2: The above analysis reveals that device-server association significantly impacts training cost in FSyL. Due to heterogeneous device capabilities and dynamic training demands, sub-optimal associations may lead to imbalanced load and increased training time. Therefore, optimizing edge association to match device demands with server resources is critical for reducing system-wide training costs.

3) *Coupled impact of container deployment and device-server association on training performance in FSyL.* It is worth noting that in multi-edge FSyL systems, container deployment and edge association jointly determine the actual set of participating devices. Specifically, container deployment defines the coverage of edge servers equipped with FSyL services, meaning that only devices within these regions are eligible for training. Additionally, effective edge association ensures that each device is matched to the most suitable nearby edge server, enabling it to complete local training and upload its update before the predefined deadline. Only those devices that successfully complete this process are considered valid participants and contribute to the global model. Consequently, device participation in FSyL is not merely a tunable parameter, but the outcome of system-level orchestration, which directly impacts model convergence and final accuracy.

The findings, depicted in Fig. 4, reveal that varying levels of device participation significantly influence the global model's

convergence speed and final accuracy. In the simple Lenet-Mnist task, although different levels of device participation achieve nearly identical test accuracy after 100 training rounds, the convergence curve under partial-device participation exhibits noticeable fluctuations during the early training period. In contrast, the complex AlexNet-Cifar10 task shows that the test accuracy drops sharply and fluctuates violently when device participation is limited. These observations align with findings reported in prior studies [27], [28], [29], showing that insufficient device participation can degrade convergence behavior and reduce model robustness.

Summary: From the above analysis, we summarize that container deployment and edge association jointly affect the system's overall training efficiency, robustness, and the number of successfully participating devices. Therefore, it is critical to design a coordinated strategy that jointly optimizes container deployment and edge association to improve training accuracy, reduce system-wide training costs, and mitigate the impact of stragglers in dynamic multi-edge FSyL.

III. SYSTEM MODEL AND PROBLEM FORMULATION

We focus on an end-edge collaborative FSyL system, which consists of a set of edge servers $\mathcal{N} = \{1, \dots, n, \dots, N\}$, each equipped with computing resources F_n and communication resources characterized by achievable bandwidth $\mathcal{W}_n$, along with a set of associated devices $\mathcal{M} = \{1, \dots, m, \dots, M\}$. Due to the limited coverage of a single edge server, we assume each server covers a different set of devices.³ We consider convolutional layers (hereafter called layers) to be the atomic elements of model structures since they dominate most FLOPs in DNNs. The DNN model is structured as a chain with a set $\{1, \dots, l, \dots, L\}$. Let f_{cl} and bc_l represent the computations required for the forward propagation and backward update of the layer l , respectively. Similarly, let fd_l and bd_l denote the IFs and IGs size of the layer l . We denote the computational capacity of device m by f_m , and the size of its local dataset by $|\mathcal{D}_m|$. The container deployment strategy profile for FSyL services is denoted as $\mathcal{B}^t = (b_1^t, \dots, b_N^t)$, where deployment strategy $b_n^t = 1$ indicates that the FSyL container is deployed at server n , otherwise $b_n^t = 0$. The deployment budget for FSyL containers is $\mathbf{B}^4$. The edge association determined by the decision $\mathcal{Z}^t = \{z_{1,n}^t, \dots, z_{m,n}^t | n \in \mathcal{N}, n \in Q_m\}$ indicates that device m is associated with the server n .

A. FSyL Deployment Benefit Model

The container deployment problem focuses on selecting appropriate edge servers for deploying containers that encapsulate the FSyL service under the given resource budgets. Assume there are N edge servers, each covering a specific wireless area and

serving a subset of devices. A device may be co-covered by multiple edge servers, and $n \in Q_m$ indicates that server n covers device m . Due to the limited service budget, the federation must select $\mathbf{B}$ servers out of the N candidates to deploy $\mathbf{B}$ FSyL services and enable E2C training. Given a set of servers where FSyL services are deployed, server n 's device coverage is denoted by $\hat{Q}_n = \{m | m \in \mathcal{M}, Q_m = n\}$. Then, the overall device coverage is measured by the total number of devices that can associate with servers equipped with FSyL services, which is defined as

$$u^c(\mathcal{B}^t) = |\bigcup_{n:b_n^t=1} \hat{Q}_n| \quad (1)$$

To further enhance the robustness of service provision and provide more alternatives for subsequent device-server association, one device is expected to be co-covered (i.e., overlapped) by multiple servers to avoid disconnection from a single server and to have more alternative servers for training robustness. The overall server co-coverage is quantified by the number of devices that are co-covered by those servers, i.e.,

$$u^r(\mathcal{B}^t) = \sum_{n:b_n^t=1} |\hat{Q}_n| - |\bigcup_{n:b_n^t=1} \hat{Q}_n| \quad (2)$$

Finally, to jointly maximize device coverage and training robustness, given the deployment strategy $\mathcal{B}^t$, the corresponding deployment benefit $DU(\mathcal{B}^t)$ can be calculated by

$$\begin{aligned} DU(\mathcal{B}^t) &= u^c(\mathcal{B}^t) + \alpha \cdot u^r(\mathcal{B}^t) \\ &= \alpha \sum_{n:b_n^t=1} |\hat{Q}_n| + (1 - \alpha) |\bigcup_{n:b_n^t=1} \hat{Q}_n| \end{aligned} \quad (3)$$

where α is the weight coefficient that balance the relative importance of device coverage and server co-coverage.

B. FSyL Association Benefit Model

1) *FSyL Cost in IRs Transmission:* The communication model describes the transmission cost of intermediate results (i.e., IFs and IGs) between the devices and the edge server. Devices upload the IFs of the cut layer to the edge server for server-side model training, and the edge server feedbacks the IGs of the cut layer to the devices for the device-side model training. Following [11], orthogonal frequency-division multiple access (OFDMA) is adopted for uplink/downlink channel access. Under OFDMA, the communication bandwidth is divided into multiple narrowband sub-channels without interference. For fairness and simplicity, we assume that the bandwidth resources are shared equally among all associated devices. Given the maximal bandwidth $\mathcal{W}_n$, the bandwidth allocated to device m by the server n is expressed as

$$r_{m,up}^t = \frac{\mathcal{W}_n}{|\mathcal{H}_n(\mathcal{Z}^t)|} \log(1 + \frac{p_m^t \delta_{m,n}^t}{\varpi_0}) \quad (4)$$

$$r_{m,dl}^t = \frac{\mathcal{W}_n}{|\mathcal{H}_n(\mathcal{Z}^t)|} \log(1 + \frac{p_n \delta_n}{\varpi_0}) \quad (5)$$

where $\delta_{m,n}$ and δ_n^t represent the uplink and downlink channel gain between the device m and the server n , and ϖ_0 represents the background noise. $\mathcal{H}_n(\mathcal{Z}^t) = \{m | z_{m,n}^t = 1, m \in \mathcal{M}\}$ represents the set of devices associated with the server n .

³A device m can be associated with an edge server n only if it is covered by n , i.e., $n \in Q_m$. Such an edge server is called device m 's neighbor server.

⁴Without loss of generality, we assume that each server's maximum available service units are equally divided, and each edge server admits at most $\mathbf{Z}$ devices to be associated with it. Therefore, after the container deployment, the total budget of edge association is $\mathbf{ZB}$. The more general cases where each server may have a different capacity $\mathbf{Z}$ are evaluated in the experiments.

Hence, the transmission cost between the device m and the associated server within one training round is expressed as

$$C_{m,com}^t = |\mathcal{D}_m| \left(\frac{fd_{pr_m}}{r_{m,up}^t} + \frac{bd_{pr_m}}{r_{m,dl}^t} \right) \quad (6)$$

2) *FSyL Cost in Local Training*: The computation process of FSyL contains two stages. First, each device m performs forward propagation until the cut layer. After receiving the IGs fed back from the server, the device m performs backward propagation to update its local model. The computation cost associated with the device-side model training is expressed as

$$C_{m,loc}^t = |\mathcal{D}_m| \frac{\sum_{l=1}^{pr_m} (fc_l + bc_l)}{f_m^t} \quad (7)$$

Then, the edge server updates the server-side model using the received IFs, and the computation cost associated with the server-side model training is expressed as

$$C_{m,ser}^t(\mathcal{Z}^t) = |\mathcal{D}_m| \sum_{l=pr_m}^L \frac{(fc_l + bc_l)}{F_n^t(\mathcal{Z}^t)} \quad (8)$$

where the computing resources provisioned to device m by the server n are expressed as $F_n^t(\mathcal{Z}^t)$. Finally, the total time cost of the device m required to accomplish the FL task within one local training round is defined as

$$C_{m,SL}^t = C_{m,loc}^t + C_{m,com}^t + C_{m,ser}^t + C_m^{sync} \quad (9)$$

If $z_{m,n}^t = 0, \forall n \in \mathcal{N}$, the total time cost of device m required to accomplish the training task within one training round is defined as

$$C_{m,FL}^t = |\mathcal{D}_m| \sum_{l=1}^L \frac{(fc_l + bc_l)}{f_m^t} + C_m^{sync} \quad (10)$$

where C_m^{sync} refers to the parameter aggregation and distribution cost of the global model.

Based on the above model analysis, if the device m is associated with an edge server n where the FSyL service is deployed and configured, the association benefit is reflected in the time cost-savings compared to training the full model locally under the standard FL mode. The total association benefit regarding time cost during round t is defined as

$$CU_m^t(\mathcal{Z}^t) = \begin{cases} C_{m,FL}^t - C_{m,SL}^t, & z_{m,n}^t = 1 \\ 0, & z_{m,n}^t = 0 \end{cases} \quad (11)$$

Finally, to maximize the overall training time cost-savings across all devices, the total association benefits under FSyL container deployment can be expressed as

$$CU^t(\mathcal{Z}^t) = \sum_{m \in \mathcal{M}} CU_m^t(\mathcal{Z}^t) \quad (12)$$

C. Problem Formulation

We aim to maximize both the deployment benefit and the association benefit by determining the FSyL container deployment during the global update stage and edge association during the local training stage, respectively. This joint optimization problem is referred to as FSyL container Deployment and edge

Association Benefit Maximization (FDAB-M).

$$\begin{aligned} \mathcal{P} : \quad & \max_{\{\mathcal{B}^t, \mathcal{Z}^t\}_{t=1}^T} \sum_{t=1}^T DU^t(\mathcal{B}^t) \cdot \mathcal{I}_{\{t\}} + CU^t(\mathcal{Z}^t) \cdot (1 - \mathcal{I}_{\{t\}}) \\ \text{s.t.} \quad & \begin{cases} \text{C1 : } b_n^t \in \{0, 1\}, \forall n \in \mathcal{N} \\ \text{C2 : } \sum_{n \in \mathcal{N}} b_n^t = \mathbf{B}, \forall t \in \mathcal{T} \\ \text{C3 : } z_{m,n}^t \in \{0, 1\}, n \in \mathcal{Q}_m^t, \forall m \in \mathcal{M} \\ \text{C4 : } \sum_{m \in \mathcal{M}} z_{m,n}^t \leq \mathbf{ZB}, \forall n \in \mathcal{N}, \forall t \in \mathcal{T} \\ \text{C5 : } \mathcal{I}_{\{t\}} = \begin{cases} 1, & t \bmod T_g = 0 \\ 0, & \text{otherwise} \end{cases} \end{cases} \end{aligned} \quad (13)$$

Constraint C1 represents whether an FSyL container is deployed on server n , and C2 indicates that the total deployment budget is $\mathbf{B}$. Constraint C3 ensures that a device may only associate with edge servers within its coverage. Constraint C4 guarantees that the total association of devices on the server does not exceed the available service capacity. $\mathcal{I}_{\{t\}}$ is an indicator function in C5. If $\{\cdot\}$ holds, then $\mathcal{I}_{\{t\}} = 1$, otherwise, $\mathcal{I}_{\{t\}} = 0$. It is noted that container deployment is invoked only during each global update phase T_g , while edge association is updated in each local training round.

D. Problem Analysis

The formulated FDAB-M problem is inherently intractable for the following reasons. **First**, the global training round T required to achieve model convergence is difficult to characterize mathematically, primarily due to the system and statistical heterogeneity. Additionally, the set of participating devices in each training round is unknown in advance, determined by the container deployment at the beginning of the global update stage T_g . As a result, the FDAB-M problem can only be addressed via a step-by-step online approach to approximate near-optimal solutions. **Second**, the device's performance is unknown before each training round due to dynamic system environments, which present challenges in accurately quantifying each device's association benefit CU^t . **Third**, it is rather challenging to provision multiple devices with both satisfied edge associations and ensure that enough devices complete local training before the deadline in each round, mainly when the service resources are limited and the training demand of the device fluctuates. In this paper, we propose an online optimization framework, named *CODEA*, to address those challenges systematically and thus provide an efficient solution to the FDAB-M problem.

IV. DESIGN OF CODEA

A. Overview of CODEA

To address the above challenges, we reformulate FDAB-M problem as an online multi-stage optimization problem within a CMAB framework, as illustrated in Fig. 5. In this framework, the system controller is regarded as an agent that interacts with the external environment, making sequential decisions to optimize training execution. Specifically, the federation observes the context information at the beginning of each round before making decisions regarding container deployment and edge association.

At the *slow-timescale* global update stage T_g , FSyL services are deployed across multiple edge servers under a fixed budget constraint to maximize deployment benefit. Subsequently, in each *fast-timescale* local training round, the system determines device-server associations on the configured servers to maximize association benefit.

Given the uncertainties and dynamics of the FDAB-M problem, *CODEA* enables the controller to make informed decisions based on real-time system states and predicted device behavior. By combining the adaptability of online learning with the precision of traditional optimization techniques, *CODEA* aims to maximize long-term rewards while generating high-quality, fine-grained deployment and association strategies.

B. Problem Transformation

In the FDAB-M problem, both the deployment and association benefits depend on multiple factors, such as device location, computing power, and transmission power, collectively referred to as the *context*. Let $x_m^t \in \mathcal{X}^t$ denote the context vector of device m observed in the t -th round, where $\mathcal{X}$ represents the context space. Without loss of generality, we assume that the context space is bounded as $\mathcal{X} = [0, 1]^D$, with D being the context dimensionality. In *CODEA*, we aim to learn and predict the association benefit of devices based on contextual information and the corresponding actual training costs. The training cost demand, mapped from the context through a specific FSyL execution mode, is a key metric to evaluate cost-efficiency. This metric reflects the cost savings achieved after the device obtains FSyL resources, defined as

$$\Psi^t(\mathcal{X}^t) = \{CU_m^t(\mathcal{Z}^t, \mathcal{X}^t) | m \in \mathcal{M}, n \in Q_m^t\} \quad (14)$$

Additionally, we define the cost-demand vector $\Psi^t = \{\Psi_m^t(x_m^t) | m \in \mathcal{M}\}$, which collects the expected cost demands of all devices under the context $\mathcal{X}^t$. In this case, the original problem can be decomposed into T independent subproblems for each training round t , as shown below

$$\begin{aligned} \max_{\{B^t, \mathcal{Z}^t\}} & \underbrace{DU^t(B^t, \Psi^t) \cdot \mathcal{I}_{\{t\}}}_{\mathcal{P}1} + \underbrace{CU^t(\mathcal{Z}^t, \Psi^t) \cdot (1 - \mathcal{I}_{\{t\}})}_{\mathcal{P}2} \\ \text{s.t. } & \text{C1} - \text{C5}, \forall t \in \mathcal{T} \end{aligned} \quad (15)$$

The above problem can be solved from the two stages of container deployment for the global update and edge association for the local training, respectively. $\mathcal{P}1$ is a subset selection problem with $\mathcal{NP}$ -hard complexity, and $\mathcal{P}2$ is an integer programming problem with knapsack constraints.

C. Submodule Maximization Container Deployment for $\mathcal{P}1$

The $\mathcal{NP}$ -hardness of $\mathcal{P}1$ is established via a polynomial-time reduction from the classical k -facility location (KFL) problem [30]. Since $\mathcal{P}1$ is conducted for each global update round, the index t is omitted for most cases in this subsection for conciseness when there is no ambiguity. To design an efficient container deployment strategy for $\mathcal{P}1$, we first prove that the set function $DU(\cdot)$ has monotone and submodular properties under the cardinality constraint $|\cdot| = B$.

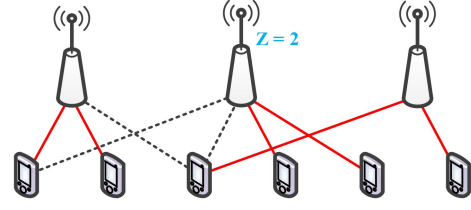


Fig. 6. An example of bipartite matching between devices and edge servers.

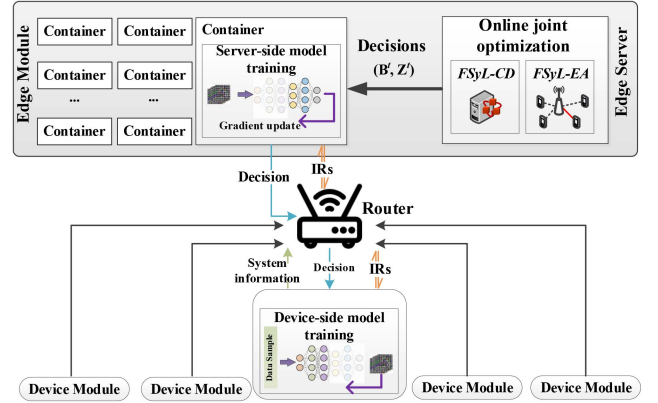


Fig. 7. FSyL system design.

Algorithm 1: Budget-Aware Container Deployment for FSyL.

- 1: **Initialize:** $B = 0, \Omega = \emptyset, DU(\Omega) = 0$
- 2: Calculate $|\hat{Q}_n|$ in parallel for all server $n \in \mathcal{N}$
- 3: **repeat**
- 4: **for** $n \in \mathcal{N}$ **do**
- 5: $DU^+(n|\Omega) = DU(\Omega \cup \{n\}) - DU(\Omega)$
- 6: $n^* \leftarrow \arg \max_{n \in \mathcal{N}} DU^+(n|\Omega), \Omega \leftarrow \Omega \cup \{n^*\}$
- 7: $b_{n^*} = 1, \mathcal{N} \leftarrow \mathcal{N} \setminus \{n^*\}, B = B - 1$
- 8: **until** $|\Omega| = B$
- 9: return container deployment strategy B^*

Definition 1 (Monotone Submodular Function). For any server subset $\Omega \subseteq R \subseteq \mathcal{N}$ and $n \in \mathcal{N} \setminus R$, the set function $DU(\cdot)$ is monotone submodular if

- $DU(\Omega) \leq DU(R)$
- $DU(\Omega \cup \{n\}) - DU(\Omega) \geq DU(R \cup \{n\}) - DU(R)$

Theorem 1 ($DU(\cdot)$ is a monotone submodular function).

The proof of Theorem 1 can be found in Appendix A, available online. After proving the monotone and submodular properties of the function $DU(\cdot)$, we propose an approximate approach FSyL-CD, i.e., Budget-aware FSyL Container Deployment, to select a subset of servers to deploy containers under budget constraints, thereby maximizing deployment benefit.

1) *FSyL-CD Algorithm Design:* The main idea of FSyL-CD is to maximize marginal utility to guide efficient container deployment in the budget-aware multi-edge scenario. Initially, we deploy budgeted FSyL services on edge servers based on the devices' physical location and servers' coverage. The details are shown in Algorithm 1.

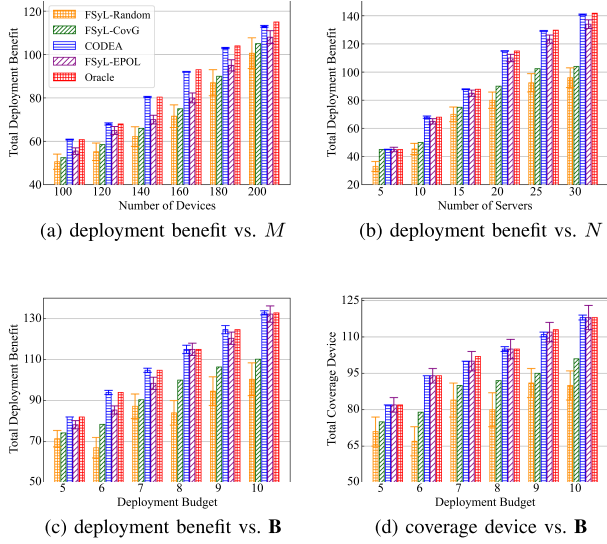


Fig. 8. Comparison of deployment benefit and device coverage under different methods.

The process of *FSyL-CD* starts with an empty set $\Omega = \emptyset$ and $DU(\Omega) = 0$, and $b_n = 0$ is assigned to each server (Line 1). Then, calculate the number of devices covered by each server $|\hat{Q}_n|$ in parallel (Line 2). From the candidate set $\mathcal{N}$, the server $n \in \mathcal{N} \setminus \Omega$ with the largest marginal gain $DU^+(n|\Omega)$ is constantly added to Ω until $|\Omega| = \mathbf{B}$ (Lines 3-8). Container deployment decisions constitute the final decision profile $\mathcal{B}^*$ as the output of the approximate algorithm. The computational complexity of the *FSyL-CD* algorithm is $\mathcal{O}(NB)$.

2) *Theoretical Analysis*: Although the proposed *FSyL-CD* is an efficient algorithm with linear computational complexity, it may not always guarantee the globally optimal solution in each global update round. Nevertheless, we can theoretically prove that *FSyL-CD* achieves a constant-factor approximation relative to the optimal solution.

Theorem 2 (*FSyL-CD achieves $(1 - e^{-1})$ -approximation guarantee for optimal container deployment benefit.*)

The proof of Theorem 2 can be found in Appendix B, available online. With this guarantee, we ensure that the container deployment in each global update stage achieves near-optimal deployment benefits, thereby establishing strong data and device support for the subsequent local training process.

D. Online Learning-Based Edge Association for P2

During the local training stage, container deployment strategy $\mathcal{B}^{*,t}$ is derived from **Theorem 1** to consider further online edge association $\mathcal{Z}^t$. To solve $\mathcal{P}2$, context-based cost demand estimation is required and combined with server coverage to formulate the edge association strategy.

1) *Overview of Online Edge Association Workflow*: We propose an online algorithm *FSyL-EA* within the CMAB framework, i.e., Federated Synergy Learning Edge Association. *FSyL-EA* follows a structured workflow in each training round. First, the federation observes the contextual information $\mathcal{X}^t = \{x_m^t\}_{m \in \mathcal{M}: m \in \hat{Q}_n^t \cap b_n^t = 1}$ along with the previously accumulated

knowledge, i.e., the correlation between contexts and cost demands). Then, edge association is performed, if $z_{m,n}^t = 1$, device m within server n 's coverage can perform *FSyL* via sub-model offloading. After completing one training round, the actual association benefit is evaluated, and the cost estimate $\Psi^t(x_m^t)$ is updated to refine future decisions.

2) *Dimensionality Reduction for Context*: In a standard MAB model, an agent plays arms actions (or arms) based on contextual empirical knowledge. However, in practical *FSyL* applications, device contexts are derived from a continuous space (e.g., computing power is a continuous variable), resulting in infinitely many possible contexts. Collecting historical demand records and learning cost-demand distributions for each possible context is both labor-intensive and impractical. To address this, *FSyL-EA* groups similar contexts and learns the demand patterns of these groups instead of each individual context $\mathcal{X}^t$. Specifically, *FSyL-EA* partitions the context space $\mathcal{X} = [0, 1]^D$ into small $(hy)^D$ hypercubes, where each hypercube is D -dimensional and has a uniform size $\frac{1}{hy} \times \dots \times \frac{1}{hy}$. These hypercubes are collected into the context partition set $\Phi \in \Phi_T$. The federation learns the cost-demand patterns across various devices. Following the Lipschitz bandit model [31], [32], arms with similar context information are expected to have similar qualities. Based on this idea, the estimated quality of the device-server pairs belonging to an identical hypercube, i.e., $\Psi_m(\Phi) = \mathbb{E}_{\varphi \in \Phi}[\Psi_m(\varphi)]$. Each device m maintains a counter $ct_{m,n}^t(\Phi)$ for each hypercube $\Phi \in \Phi_T$ up to round t , recording the number of times that device m has associated with a nearby server (i.e., $z_{m,n}^t = 1, x_m^t \in \Phi$).

3) *FSyL-EA Algorithm Design*: Algorithm 3 presents the pseudocode for *FSyL-EA*. At the t -th round, the federation observes the contextual information $\mathcal{X}^t = \{x_m^t\}_{m \in \mathcal{M}}$ and determines the hypercube Φ to which each device belongs, namely $x_m^t \in \Phi_m^t$ (i.e., dimensionality reduction for context). The hypercubes for all devices are collected in $\Phi^t = \{\Phi_m^t\}_{m \in \mathcal{M}}$. The estimated cost demand for device m during round t is obtained via $\Psi_m(\Phi_m^t)$ (note that x_m^t has been replaced by the hypercube Φ_m^t). The cost estimates are collected in $\Psi^t = \{\Psi_1(\Phi_1^t), \dots, \Psi_M(\Phi_M^t)\}$. *FSyL-EA* operates in either an exploration or exploitation phase. To determine the phase for the current round, the algorithm checks whether the current contexts of the devices have been sufficiently explored. We define the set of under-explored devices Π^t based on the contexts observed and counters in round t as follows:

$$\Pi^t(\mathcal{X}^t) = \{m \in \mathcal{M} | ct_{m,n}^t(\Phi_m^t) < K(t), x_m^t \in \Phi_m^t\} \quad (16)$$

where $K(t)$ is a monotonically increasing control function used to determine whether the amount of historical data collected in hypercube Φ_m^t is well-explored enough to accurately predict cost demand for exploitation.

Exploration Stage: If the under-explored set is not empty, i.e., $\Pi^t \neq \emptyset$, *FSyL-EA* enters the exploration stage. During this stage, the goal is to prioritize the association of under-explored devices to gather more context-reward information. Before that, we should identify the maximum under-explored device-server pairs (m, n) since each device can only be associated with one

Algorithm 2: Maximum Device-Server Pair Exploration.

```

1: Initialize:  $\mathcal{M}' \in \Pi^t, \mathcal{N}' \in \mathcal{Q}_{\mathcal{M}'}, \mathbf{E}, \mathbf{Z}$ 
2:  $m \leftarrow |\mathcal{M}'|, n \leftarrow |\mathcal{N}'|$ 
3:  $k \leftarrow [0]_{(n+1) \times (\mathbf{Z}+1)}, \Xi \leftarrow [0] * m$ 
4: for  $i \in m - 1$  do
5:   for  $j \in n - 1$  and  $j \in \mathbf{E}[i]$  do
6:     for  $w \leftarrow \mathbf{Z}, \dots, 1$  do
7:       if  $\Xi[i] = 0$  &  $k[j][w] < k[j][w - 1] + 1$  then
8:          $k[j][w] \leftarrow k[j][w - 1] + 1$ 
9:          $\Xi[i] \leftarrow 1, G[i] \leftarrow j$ 
10: return device-server association pair  $G$ 

```

server in each training round (although it may have multiple neighboring servers $\mathcal{Q}_m$).

Definition 2 (Maximum Under-explored Pairs). Given the set of under-explored devices Π^t , the maximum set of under-explored device-server pairs (m, n) is denoted as Ω .

We define the device-server mapping relationship as $\mathbf{E} = \{(m, n) | m \in \Pi^t, n \in \mathcal{Q}_m\}$. The maximum under-explored pairs Ω can be transformed into a Multi-Constrained Bipartite Matching Problem (MCBMP), as shown in Fig. 6. To solve this maximum matching problem, we can refer to the spirit of Edmonds-Karp [33], which combines breadth-first search (BFS) and residual network construction to efficiently find the shortest augmenting paths and increase the matching flow along these paths iteratively. By repeatedly augmenting along the shortest paths, the maximum matching for the bipartite graph can be effectively obtained. The algorithm ensures the optimal solution and has the time complexity of $\mathcal{O}((M + N + 2)E^2)$, where E denotes the number of links.

There are two cases: *i)* If $|\Omega| \geq \mathbf{ZB}$, *FSyL-EA* can only explore a subset of devices from Π^t . Intuitively, we want to gather more information regarding the cost demands of under-explored devices. However, since the exact cost demands of these devices are unknown, the exploration stage of *FSyL-EA* reduces to solving a Constrained Multiple Knapsack Problem (CMKP) with server coverage constraints. The objective is to maximize the number of associated devices, thereby expanding exploration coverage. *FSyL-EA* adopts dynamic programming to solve the maximization exploration problem.

In the exploration stage, the objective is to select as many under-explored devices as possible without exceeding the total association budget $\mathbf{ZB}$. To achieve this, Algorithm 2 maximizes the number of devices explored under resource constraints. The algorithm ensures the optimal solution with a time complexity of $\mathcal{O}(\mathbf{ZMN})$. *ii)* if $|\Omega| < \mathbf{ZB}$, *FSyL-EA* select all under-explored devices in the set Π^t , and there are $\mathbf{ZB} - |\Omega|$ remaining budget. The remaining budget determines the edge association decision on the explored devices $m \in \mathcal{M} \setminus \Pi^t$ based on the current cost-benefit estimates. The resulting edge association decisions $\mathcal{Z}^t$ are obtained as follows:

$$\begin{aligned} \max_{\mathcal{Z}^t} \quad & CU^t(\mathcal{Z}^t, \Psi^t) \\ \text{s.t.} \quad & m \in \mathcal{M} \setminus \Pi^t, C3 - C5 \end{aligned} \quad (17)$$

Algorithm 3: CODEA for FSyL.

```

1: Initialize:  $\mathcal{M}, \mathcal{N}, hy, ct_{m,n}(\Phi) = 0, \Psi_{m,n}(\Phi) = 0$ 
2: for  $t = 1, \dots, T$  do
3:   Observe the context  $\mathcal{X}^t$ ; Determine content partitions
    $\Phi^t = \{\Phi_m^t\}_{m \in \mathcal{M}}$ , where  $\Phi_m^t \in \Phi_T$  such that
    $x_m^t \in \Phi_m^t$ 
4:   if  $t \bmod T_g = 0$  then
5:     Execute Algorithm 1  $\triangleright$  Container Deployment
6:   else
7:     Execute Device-Server Edge Association
   // Device-Server Edge Association:
8:   Identify the under-explored set  $\Pi^t \triangleright$  Exploration
9:   Calculate the maximum under-explored pairs  $\Omega^t$ 
10:  if  $\Pi^t \neq \emptyset$  then
11:    if  $|\Omega| \geq \mathbf{ZB}$  then
12:      Select under-explored pair  $G^t \subseteq \Pi^t$  according to
      Algorithm 2
13:       $z_{m,n}^t = \mathbf{1}_{\{(m,n) | (m,n) \in G^t\}}$ 
14:    else  $\triangleright$  i.e.,  $|\Omega| < \mathbf{ZB}$ 
15:      Solve for  $\mathcal{Z}^t$  using (17)
16:    else  $\triangleright$  Exploitation
17:      Solve for  $\mathcal{Z}^t$  using (18)
18:  If  $z_{m,n}^t = 1$ , observe actual cost demand.  $\triangleright$  Update
19:  for each device  $m$  with  $z_{m,n}^t = 1$  do
20:     $\Psi_{m,n}(\Phi_m^t) = \frac{\Psi_{m,n}(\Phi_m^t)ct_{m,n}(\Phi_m^t) + CU_m}{ct_{m,n}(\Phi_m^t) + 1}$ 
21:     $ct_{m,n}(\Phi_m^t) = ct_{m,n}(\Phi_m^t) + 1$ 
22: return  $\mathcal{Z}^t, t = 1, 2, \dots, T$ 

```

Exploitation Stage: If $\Pi^t = \emptyset$, candidate device-server pairs have been explored enough, then *FSyL-EA* enters the exploitation stage. The federation determines the optimal edge association decision according to the cost demand estimate Ψ^t . The association decision $\mathcal{Z}^t$ can be obtained as follows:

$$\begin{aligned} \max_{\mathcal{Z}^t} \quad & CU^t(\mathcal{Z}^t, \Psi^t) \\ \text{s.t.} \quad & C3 - C5 \end{aligned} \quad (18)$$

Update Stage: After the exploration and exploitation stages, the federation updates the context-cost estimation according to participating devices' actual observed cost demands.

4) *Theoretical Analysis:* Regret is a key performance metric that quantifies the gap between a given policy and the optimal policy, and is used to evaluate the convergence behavior of online algorithms [34], [35]. Finally, we theoretically analyze the learning regret upper bound of the proposed *FSyL-EA*.

Definition 3 (Regret Upper Bound). Assume that the federation knows the expected association cost demands precisely under each context. In this case, the original problem can obtain an Oracle solution for each training round t . The cumulative regret of the proposed *FSyL-EA* is defined as:

$$\mathcal{R}(T) = \sum_{t=1}^T (\mathbb{E}[CU^t(\mathcal{Z}^{*,t}, \Psi^t)] - \mathbb{E}[CU^t(\mathcal{Z}^t, \Psi^t)]) \quad (19)$$

Theorem 3 (Regret Upper Bound of FSyL-EA) Let $hy = [T^{\frac{1}{3\delta+D}}]$, $K(T) = T^{\frac{2\delta}{3\delta+D}} \log(T)$ and $\Gamma = \mathbf{ZB}^2 MC_{\max}$. If

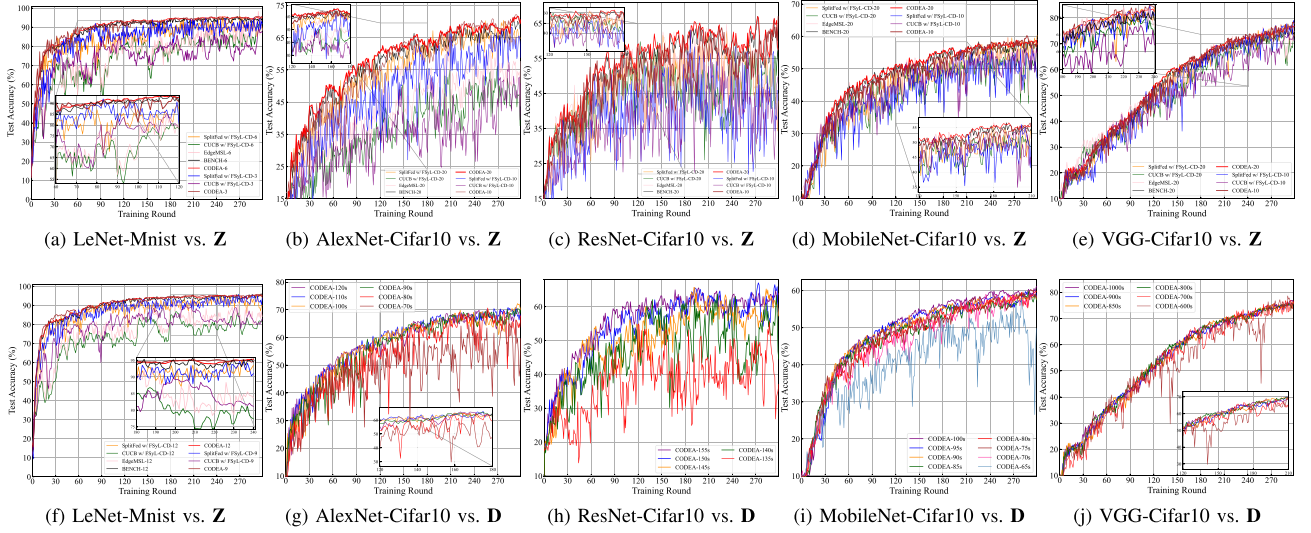


Fig. 9. Comparison of training performance under different methods and different configurations.

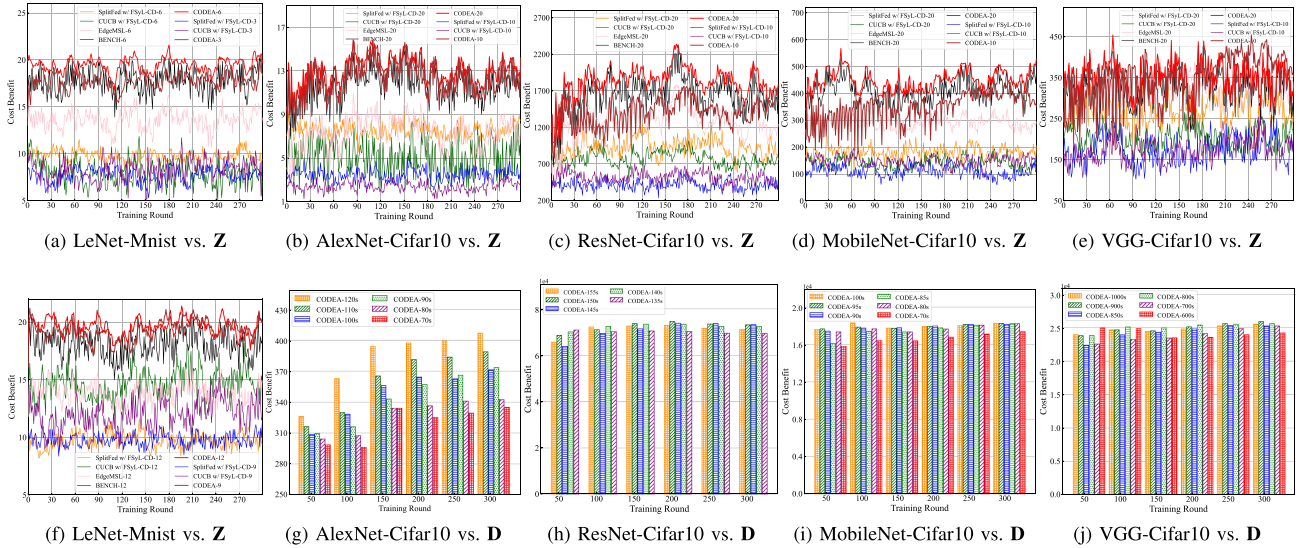


Fig. 10. Comparison of cost benefit under different methods and different configurations.

FSyL-EA is employed with these parameters and the Hölder Condition holds true, the regret $\mathcal{R}(T)$ is bounded by

$$\begin{aligned} \mathbb{E}[\mathcal{R}(T)] &\leq \Gamma 2^D (\log(T) T^{\frac{2\theta+D}{3\theta+D}} + T^{\frac{2}{3\theta+D}}) \\ &\quad + \Gamma N \sum_{k=1}^{\mathbf{ZB}} \binom{M}{k} \frac{\pi^2}{3} \\ &\quad + \Gamma N (3\mathcal{L}2^{\frac{\theta}{2}} + 2 \frac{(1 + \mathcal{L}2^{\frac{\theta}{2}})(3\theta + D)}{(2\theta + D)}) T^{\frac{2\theta+D}{3\theta+D}} \end{aligned}$$

The proof of **Theorem 3** is in Appendix C, available online. The leading order of the regret $\mathcal{R}(T)$ is $\mathcal{O}(\mathbf{ZB}^2 M C_{\max} 2^D T^{\frac{2\theta+D}{3\theta+D}} \log(T))$, which is sublinear in the time horizon T .

V. EXPERIMENT

In this section, we conduct comprehensive experiments and comparative analysis within a scalable FSyL simulated system to evaluate the training performance of the proposed mechanism. Specifically, we first briefly introduce the experimental setup, including system parameters and comparative methods. Then, we evaluate the influence of the task type, device population, server scale and service budget on the FSyL benefit, training efficiency, test accuracy, and algorithm efficiency. Finally, evaluation results are presented and discussed.

A. Experiment Settings

We evaluated the training performance of the proposed mechanism through experiments conducted in a simulated FSyL environment, as illustrated in Fig. 7. Our evaluation is

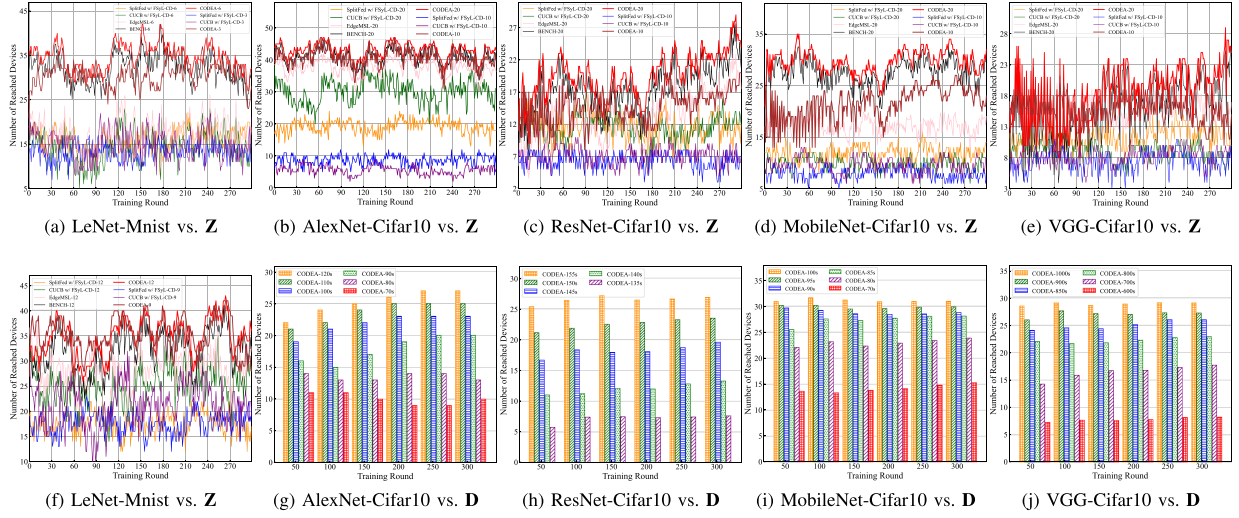


Fig. 11. Comparison of the number of participating devices under different methods.

conducted on the physical locations of real-world base stations and end-users within Metropolitan Melbourne in Australia, which records massive information of edge servers and end users [26], [36]. We chose a dataset that included 30 server records and 200 user records. The locations of servers and users, as well as the computing/communication capabilities and data volume, are regarded as context. The level of heterogeneity is controlled by computation frequency and transmission bandwidth [7], [27]. We assume that the edge server's computing capability F_n and wireless bandwidth $\mathcal{W}_n$ are 10GHz and 5MHz, respectively. The background noise ϖ_0 is -100dBm , and the noise power channel gain $\sigma_{n,m}$ is $[0.01, 0.05]$. The available computation capacity f_m and transmission power p_m are sampled from the same uniform distribution within $[0.5, 2]\text{GHz}$ and $[0.05, 0.18]\text{W}$, respectively. All evaluations run on the same hardware, and the runtime is averaged over 300 global rounds (matching the main experiment's horizon) to smooth short-term variability.

Model and Training Dataset: We evaluate our mechanism across diverse DNNs, including LeNet-5, AlexNet, ResNet-18, MobileNet, and VGG-9 for vision, and a character-level CNN (CharCNN) for NLP. Image tasks utilize MNIST and CIFAR-10, while text tasks employ Yahoo Answers (10 classes) and DBPedia (14 classes). For CharCNN, we follow the standard character alphabet (approximately 70 symbols) and fix the input length to 1014 by truncating or padding. To simulate a realistic FSyL scenario, we induce non-IID client data, i.e., each device holds samples from two unique labels for vision, and adopt a Dirichlet distribution for NLP.

Comparison Benchmarks: We implement the following methods as benchmarks, with a brief description of each comparison method provided below.

- **Oracle [37]:** Oracle denotes optimal container deployment and edge association, which builds the model presented in Section IV-B and employs IBM's CPLEX Optimizer to solve the integer programming model for individual rounds optimally.

- **FSyL-CovG [38]:** The idea of *FSyL-CovG* comes from the FL aggregators deployment, aiming to ensure the most covered participating devices [11]. The vanilla *FSyL-CovG* is designed for FL, so the co-coverage benefit of servers and device-server association is not considered.
- **FSyL-Random:** *FSyL-Random* randomly selects a set of servers to deploy FSyL containers with budget constraints, which is an adaptation to multi-edge SFL [24].
- **FSyL-EPOL [39]:** *FSyL-EPOL* formulates model deployment as a potential game and uses best-response dynamics to converge to a near-optimal equilibrium, aiming to maximize the user coverage.
- **SplitFed w/ FSyL-CD [21]:** Online edge association is determined based on the proposed *FSyL-CD*. Note that we have adapted SplitFed to accommodate our context, but the basic idea is the same as the vanilla one, which is to associate with different servers. Each device randomly associates with the server in the coverage area without prior knowledge. Given its simplicity, random device association can be applied to any FL scenario.
- **CUCB w/ FSyL-CD [35]:** On the basis of *SplitFed w/ FSyL-CD*, the classical MAB policy UCB is integrated. CUCB first enumerates all feasible edge association decisions and runs UCB with feasible decisions $z_{m,n}$, and learns the expected utility for each device-server pair (m, n) in the edge aggregation round. Because of this, CUCB is only suitable for the general-scale edge association problem, and the large-scale scenario will make the decision space explode exponentially.
- **BENCH [40]:** On the basis of *SplitFed w/ FSyL-CD*, *BENCH* prioritizes maximizing the number of device-server links under coverage and capacity constraints. Concretely, *BENCH* favors breadth of participation over per-link latency or utility.
- **Edge-MSL [41]:** On the basis of *SplitFed w/ FSyL-CD*, *Edge-MSL* computes UCB-style indices for each device-server pair and then solves a global assignment under

capacity and reachability constraints, yielding a collision-free edge association in each round that improves on-time successful updates. This centralized matching affords effective resource coordination but incurs nontrivial per-round complexity, making it most suitable for moderate-scale scenarios.

B. Performance Comparison

Test Case 1 (Deployment Benefit): To evaluate whether *CODEA* could effectively improve deployment benefit and device coverage, in this case, we evaluate the performance of various methods in the multi-edge FSyL scenario. As shown in Fig. 8, the results show the performance of various container deployment strategies regarding total deployment benefit and device coverage under various configuration parameters. In general, *CODEA* consistently outperforms both *FSyL-CovG*, *FSyL-EPOL* and *FSyL-Random*, and achieves performance comparable to that of the *Oracle* in multiple experimental settings. Specifically, Fig. 8(a) shows the superior performance of *CODEA* when the number of servers n is 20 and the number of devices m increases from 100 to 200. While *Oracle* always achieves the upper bound, the performance gap between *CODEA* and *Oracle* narrows as the number of devices grows. This is attributed to higher user density, which increases the probability of device co-coverage, thus enhancing deployment flexibility. Overall, across all cases, *CODEA* outperforms *FSyL-CovG* and *FSyL-Random* significantly.

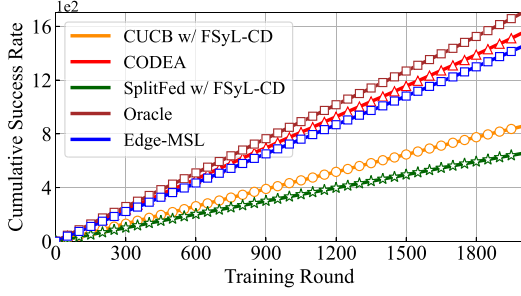
Fig. 8(a)–(c) further confirm that *CODEA* consistently maintains the highest deployment benefit across three scenarios. As the number of devices increases from 100 to 200, *CODEA* consistently achieves the highest deployment benefit, demonstrating superior adaptability and scalability. Within the 5 to 30 server quantity range, *CODEA* continues to lead in utility growth. Notably, as the number of servers increases, *FSyL-CovG* and *FSyL-Random* struggle to effectively leverage co-coverage due to their lack of robustness in server selection, and the probability of the *FSyL-Random* making mistakes becomes greater. In contrast, *CODEA* demonstrates strong adaptability by incorporating both coverage and robustness objectives into deployment decision-making. Moreover, within the 5 to 10 deployment budget range, *CODEA* maintains its advantage in both deployment benefit and device coverage, particularly at higher budget levels. This also provides sufficient dataset resources for subsequent local training.

Test Case 2 (Test Accuracy): To evaluate whether *CODEA* ensures model convergence and accuracy improvement, we evaluate the convergence rate and test accuracy of the global model under different experimental configurations (i.e., the maximum number of associated devices supported by each server Z and the deadline D for each training round). For better readability, enlarged versions of the figures are provided in the Appendix, available online. In Fig. 9(a) and (f), we measured the test accuracy of the global model while the Z is set to $\{3, 6, 9, 12\}$. Results show that *CODEA* achieves efficient convergence and superior accuracy in the LeNet-Mnist task compared with the benchmarks.

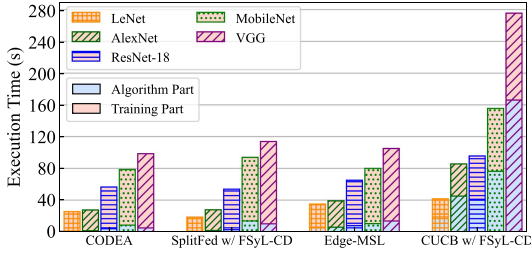
Although benchmarks (particularly for small-structure models) remain feasible, *CODEA* consistently maintains the most stable and highest training performance, especially during the early training stage. Fig. 9(b) and (g) demonstrate that *CODEA* maintains clear advantages in the more complex AlexNet-Cifar10 task, with Z and D are set as $\{10, 20\}$ and $[70 - 120]$ s, respectively. Although *BENCH* strives to maximize the number of associated devices, it fails to consider the inter-device heterogeneity, resulting in a suboptimal solution. Compared with the benchmark methods, *CODEA* improves model accuracy by up to 20% and achieves faster convergence by accelerating training and maximizing the collection of local updates before the deadline. Moreover, consistent convergence behavior observed across various model architectures, including ResNet18, MobileNet, and VGG9, demonstrates that *CODEA* generalizes well to diverse DNN structures, confirming its robustness and applicability. In summary, *CODEA* effectively improves training efficiency and model accuracy by leveraging precise demand estimation and adaptive device-server association.

Test Case 3 (Cost Benefit): To evaluate whether *CODEA* achieves efficient training toward time cost savings, we tested 300 training rounds and calculated per-round association benefits under the same experimental settings as in the above test case. Fig. 10 illustrates the effectiveness of *CODEA* and the impacts of two parameters, i.e., service resources and deadline constraints. Overall, *CODEA* consistently achieves the highest cost benefits across all scenarios. Fig. 10(a) and (f) show the results for simple model structures. Benchmarks (*SplitFed w/ FSyL-CD* and *CUCB w/ FSyL-CD*) show inferior performance, particularly *SplitFed w/ FSyL-CD*. The reason is that all devices adopt the same association decision probability to connect with neighboring servers and share the limited service resources equally. As a result, some weak devices may fail to associate with nearby servers and fall back to FL-mode updates, which leads to intolerable training delays. Meanwhile, even strong devices are associated with nearby servers but have not achieved satisfactory cost savings, and the reduction of training time is limited. Compared to the benchmarks, *CODEA* achieves the highest association benefit across different configurations. The reason is that devices hardly guarantee the training efficiency of complex models, and the training improvement brought by edge association is very prominent. Results show that *CODEA* increases up to $3\times$ and $1.5\times$ association benefit under different configurations compared with benchmarks. We conclude that context-aware online edge association leads to optimal training performance and substantial time cost savings.

Test Case 4 (Device Participation): To evaluate the effectiveness of *CODEA* in facilitating efficient FSyL under deadline constraints, we measured the number of devices that successfully uploaded their local model updates across different methods. As shown in Fig. 11, *CODEA* consistently outperforms all baselines. Specifically, *SplitFed w/ FSyL-CD* exhibits significant inefficiency in heterogeneous multi-edge scenarios due to device diversity in computation and communication capabilities. Many devices fail to complete local training within the predefined deadline, leading to severe underutilization of system resources and low participation rates. Although *CUCB w/ FSyL-CD*



(a) Cumulative Success Rate



(b) Execution Time

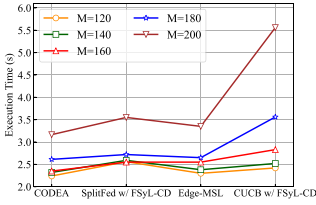
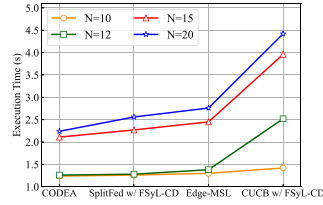

 (c) Deployment Time vs. M

 (d) Deployment Time vs. N

Fig. 12. Comparison of the effectiveness of different methods.

leverages a contextual bandit approach, its short-sighted optimization yields suboptimal association decisions. It struggles to balance exploration and exploitation, leading to inefficient resource usage and an update success rate below 50%. In contrast, *CODEA* employs multi-stage online learning to adaptively balance short-term cost estimation and long-term benefit. Incorporating contextual device information and historical cost patterns enables robust device-server associations that account for system heterogeneity and workload dynamics. Across all test scenarios, *CODEA* consistently achieves the highest success rate in timely update collection, maximizing effective device participation and accelerating model convergence.

Test Case 5 (Algorithm Effectiveness): We further calculate the cumulative success rate of collecting local updates, which is a crucial metric to evaluate the long-term performance of online learning algorithms. Fig. 12(a) demonstrates the cumulative learning reward measured by the ratio of successfully collected local updates obtained by different methods. As expected, the *Oracle* achieves the highest cumulative success rate, which gives an upper bound to the other algorithms. In contrast, baseline methods such as *SplitFed w/ FSyL-CD* and *CUCB w/ FSyL-CD* exhibit slower growth due to their limited

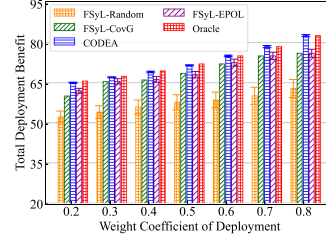
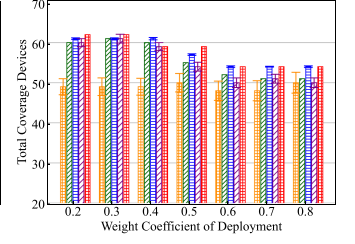
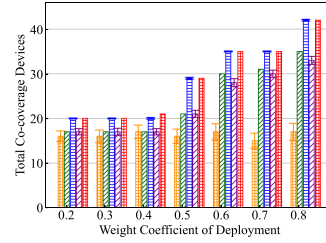
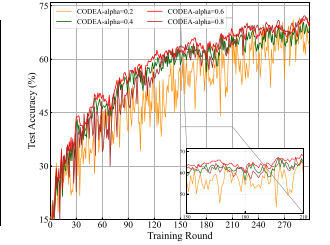

 (a) Deployment benefit vs. α

 (b) Coverage devices vs. α

 (c) Co-coverage devices vs. α

 (d) End-to-end training vs. α

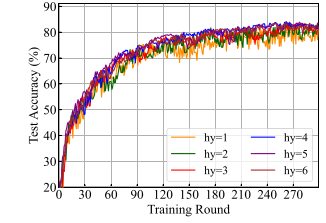
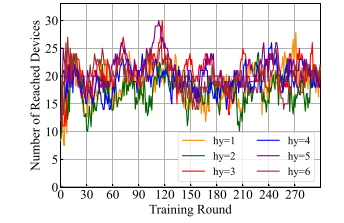
 Fig. 13. Sensitivity analysis under different weight α .

 (a) Training performance vs. hy

 (b) Participating devices vs. hy

 Fig. 14. Sensitivity analysis under different partition hy .

adaptability in optimizing device-server association under heterogeneous and dynamic systems. However, *CODEA* exhibits a quasi-linear growth trend in cumulative reward as training progresses. This indicates that *CODEA* effectively balances the trade-off between exploration and exploitation, gradually refining its understanding of the cost-performance characteristics of devices. Complementary to the reward curves, Fig. 12(b)–(d) report the per-round container deployment time and training execution time across different methods. *CODEA* maintains a low and stable runtime comparable to *SplitFed w/ FSyL-CD*, while maintaining competitive training execution time. Overall, *CODEA* consistently outperforms all benchmarks (including *Edge-MSL*) and achieves a close-to-Oracle cumulative utility, showcasing its superior adaptability and learning efficiency in multi-edge FSyL scenarios.

Test Case 6 (Sensitivity Analysis of Hyperparameters):

1) Coverage/Co-coverage Weight α : We sweep $\alpha \in \{0.2, 0.3, \dots, 0.8\}$ under a fixed budget and report total deployment benefit and total covered/co-covered devices across five schemes, as shown in Fig. 13(a)–(c). As α increases (i.e., co-coverage is valued more), the overall deployment benefit rises steadily, driven by an apparent, near-monotonic increase in co-covered devices. In contrast, the number of uniquely

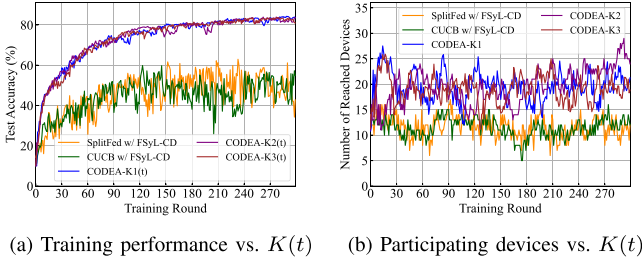


Fig. 15. Sensitivity analysis under different threshold functions $K(t)$.

covered devices shows a mild non-increasing trend. The ranking among methods is stable across α , with *FSyL-EPOL* and *CODEA* consistently outperforming *FSyL-Random* and *FSyL-CovG*, indicating that explicitly accounting for overlap yields higher utility when redundancy is rewarded. Practically, setting $\alpha \in [0.5 - -0.7]$ strikes a good balance between breadth (coverage) and resilience (co-coverage), whereas larger α prioritizes robustness and load-balancing at the expense of marginal coverage.

To further verify the influence of α on end-to-end training performance, we sweep $\alpha \in \{0.2, \dots, 0.8\}$ under a fixed budget and run the full *CODEA* pipeline across multiple rounds. The curves in Fig. 13(d) show a consistent pattern: small α (e.g., $\alpha = 0.2$, coverage-dominant) yields faster early gains because more distinct devices are covered, but the curve exhibits larger variance. Since greedy coverage provides limited association options, deadline misses and unstable associations accumulate as load fluctuates. A very large α (e.g., $\alpha = 0.8$, strongly co-coverage-oriented) improves robustness but narrows the spatial reach and slightly slows early accuracy growth. A mid-high setting ($\alpha \approx 0.6$) best balances the two, i.e., it preserves broad enough coverage while ensuring sufficient co-coverage to reassign devices when links/loads change, producing the most stable trajectory and highest final accuracy among the tested values. These trends persist across different budgets and device scales, indicating that a moderate emphasis on co-coverage is the most effective solution.

2) *Context Partition Granularity hy* : We sweep $hy \in \{1, 2, 3, 4, 5, 6\}$ and track test accuracy and the number of reached devices per round. As shown in Fig. 14(a)–(b), across all settings, the learning curves share a similar rising trend and converge to a comparable plateau, indicating that our association learner is generally robust to the partition resolution. However, coarser partitions $hy \in \{1, 2\}$ yield noisier trajectories and a slightly lower final accuracy, together with larger fluctuations in per-round participation. This reflects the fact that coarse hypercubes pool heterogeneous contexts, weakening the learner’s ability to discriminate costs/rewards and thus degrading stability and peak accuracy. In contrast, moderate to fine partitions $hy \in [4 - 6]$ strike a better balance: they preserve most of the efficiency benefits while maintaining smoother training dynamics and higher asymptotic accuracy. Practically, we recommend a mid-range hy (e.g., $hy = \{4, 5\}$) as it achieves near-best accuracy with limited overhead, whereas very coarse hy improves

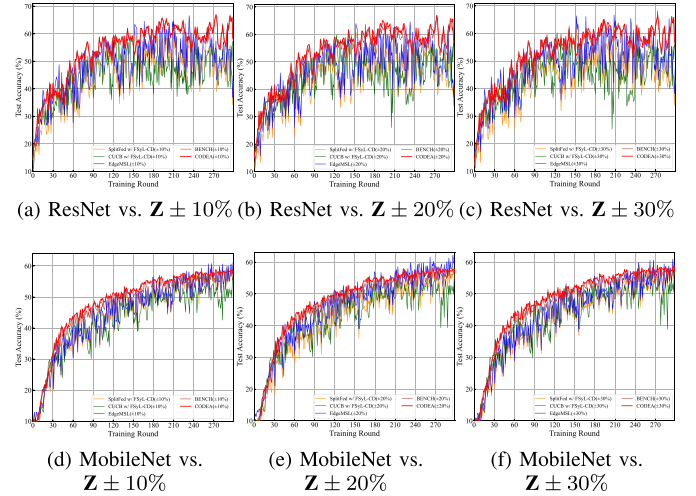
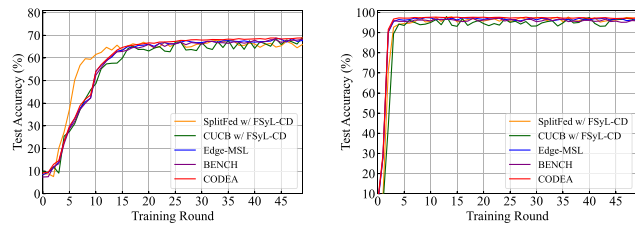


Fig. 16. Performance analysis under variable-capacity $\mathbf{Z}$.

lookup/estimation efficiency at the expense of precision and stability.

3) *Exploration Threshold Function $K(t)$* : We benchmark three exploration schedules, i.e., $K_1(t) = \sqrt{t \log t}$, $K_2(t) = t^{\frac{2}{2+\beta}} \log t$, and $K_3(t) = t/\log t$, under the same deployment budget and deadline settings. As shown in Fig. 15(a)–(b), all *CODEA* variants exhibit broadly similar learning trajectories and consistently outperform *SplitFed w/ FSyL-CD* and *CUCB w/ FSyL-CD* baselines in both final accuracy and the number of on-time participating devices. Among the three exploration functions, $K_2(t)$ achieves the best exploration–exploitation balance: it converges slightly faster to the high-accuracy regime and sustains a higher reached-device rate across training rounds. Intuitively, $K_2(t)$ and $K_1(t)$ inject sufficient early exploration to avoid premature commitment, while preventing the runaway exploration of $K_3(t)$ that can depress short-term participation and slow accuracy gains. Overall, these results indicate that moderate exploration growth (i.e., our default $K_2(t)$) yields efficient learning without sacrificing stability.

4) *Server Capacity Constraint $\mathbf{Z}$* : We further evaluate a variable-capacity setting in which each server’s admissible load fluctuates by $\mathbf{Z} \pm \{10\%, 20\%\}$ around its nominal limit, and re-run the end-to-end *FSyL* training. As shown in Fig. 16, across these dynamics, *CODEA* consistently reduces training time, and the qualitative trends remain unchanged relative to the fixed-capacity case. Compared with the fixed-capacity setting, the convergence trajectories exhibit slightly larger fluctuations, but these remain within acceptable tolerance. We further extended the evaluation to consider higher levels of heterogeneity (i.e., $\mathbf{Z} \pm 30\%$ capacity fluctuations). The results demonstrate that *CODEA* maintains its advantage over baseline methods even in scenarios with significant capacity variability. While fluctuations in convergence trajectories become more pronounced as heterogeneity increases, *CODEA* still exhibits graceful performance degradation, ensuring robustness in dynamic, heterogeneous edge environments. These experiments provide stronger support for the scalability and adaptability of *CODEA* in real-world, resource-constrained edge scenarios.



(a) Training performance vs. Yahoo (b) Training performance vs. DBpedia

Fig. 17. Comparative analysis of training performance for NLP tasks.

Test Case 5 (Usefulness in NLP Tasks): We evaluate *CODEA* on NLP tasks using CharCNN and text classification benchmarks (Yahoo and DBpedia). As shown in Fig. 17, *CODEA* outperforms baselines, i.e., *SplitFed w/ FSyL-CD*, *CUCB*, and *Edge-MSL*, demonstrating strong cross-domain performance. *CODEA* achieves higher local update success rates by dynamically adjusting service resources through multi-edge container deployment and context-aware device-server association, reducing training latency and improving participation without sacrificing accuracy. The results highlight *CODEA*'s capability to maintain high test accuracy and device participation across diverse data sets, reinforcing its effectiveness in managing resources in both vision and NLP tasks. These findings emphasize the versatility of *CODEA* in real-world edge computing scenarios.

To summarize, the proposed *CODEA* can significantly enhance training performance, delivering higher training accuracy, expedite model convergence, reduce training cost, lower computational complexity, and strong cross-domain adaptability, thereby providing practical guidance for efficient FSyL in heterogeneous and resource-constrained edge environments.

VI. RELATED WORK

In this section, we introduce existing efforts related to FL and SFL on the edge side. These works mainly focus on optimizing the training procedure to accelerate training task completion and enhance resource efficiency.

Federated Edge Learning: Recently, FEEL has emerged as a mainstream collaborative framework in edge learning, driving extensive research efforts toward enhancing its performance. Tran et al. [7] explored the impact of heterogeneous constraints on training costs by formulating a trivariate optimization problem to balance training time and energy cost. HCEF jointly optimizes update frequency and compression ratio to handle device heterogeneity in dynamic FEEL systems, achieving lower training time and energy consumption while maintaining high accuracy [42]. FedCG addresses communication efficiency, network dynamics, and device heterogeneity, achieving significant speedups compared to baselines [43]. Wu et al. [11] presented a joint edge aggregation and association mechanism, which focuses on optimizing resource allocation and communication overhead, thereby improving the overall FL efficiency. Xu et al. optimized training costs by jointly configuring edge aggregation node placement and participating user allocation for FL requests, considering the model uncertainty [12] and user dynamics [44],

respectively. Yang et al. [8] proposed a fairness-aware incentive mechanism for multi-server FL, integrating differential privacy protection. Formulating the server-node interaction as a Stackelberg game derives optimal strategies via backward induction, ensuring efficient participation while balancing privacy-utility trade-offs. Despite these advancements, the above works ignore the bottleneck of training large-scale DNN models in the resource-constrained edge computing environment [45]. This oversight highlights a critical area for future research in SFL.

Split Federated Learning: Recent advancements in SFL have explored various methods to optimize training efficiency, privacy protection, and resource management in resource-constrained and heterogeneous edge environments. The SplitFed framework, proposed by Thapa et al. [21], aims to offer a higher level of privacy and better efficiency compared to traditional FL [46], [47]. Deng et al. [48] introduced a hybrid split and federated learning framework that employs a lightweight model decomposition method to balance training latency and privacy leakage. The edge learning cost and training accuracy are optimized through multi-dimensional game-theoretical approaches, including model splitting, resource provisioning, and device association in a single server scenario [27]. Jiang et al. [16] introduced FedSyL, which balances training efficiency and data leakage risk by predicting training latency and optimizing sub-model offloading for heterogeneous devices. EPSL developed an efficient parallel split learning method, emphasizing the importance of optimal device-side sub-models to address computation heterogeneity, thereby reducing training latency and improving system efficiency [20]. Huang et al. [49] explored a decentralized method of proactive model offloading and resource management in SFL, enhancing training efficiency and reducing latency through decentralized strategies. HSFL combined SL and hierarchical FL to improve the training latency and execution energy. HSFL employed a BSUM-based method for the combined relative local accuracy minimization, association, resource, and transmit power allocation [50]. Hourglass optimized SFL by minimizing GPU storage and computation via dynamic model partitioning and feature clustering, achieving faster convergence and higher accuracy in practice [14].

Related works on resource management and edge association of multi-edge scenarios were discussed in [22], [23], [24], [25], [41], [51], [52], a parallel SL framework and a two-timescale resource management scheme were proposed to improve training efficiency [22]. ARES considered resource-constrained devices and time-varying network throughput to balance training efficiency and execution cost in multi-edge SFL [23]. Cheng et al. [25] proposed a CHEESE framework for delay-efficient FL in a D2D network, which leverages the computation and communication resources of helping devices to alleviate training costs of learning devices. Tirana et al. [51] investigate workflow optimization for parallel SL, providing device-helper assignments and scheduling strategies to enhance training efficiency in distributed edge systems. Lin et al. [52] propose a hierarchical SFL framework to address the challenges of deploying large AI models on resource-constrained devices within multi-tier cloud-edge systems, which establishes a theoretical convergence bound and develops a joint model splitting and aggregation

method to enhance SFL performance. However, these works assume that each server is pre-configured with the FL or SFL services, neglecting budget-aware container deployment, adaptive resource adjustment, and online edge association in multi-edge scenarios.

In a nutshell, the fundamental limitation of existing works is ignoring the influence of budget-constrained service provisioning and device-server edge association on the FSyL training performance. Most prior works assume idealized environments with abundant resources and static associations, overlooking real-world systems' dynamic and heterogeneous nature. Unlike the above works, we focus on designing an online budget-aware container deployment and context-aware edge association scheme that tackles user dynamics and system heterogeneity to enable efficient and cost-effective training in a scalable multi-edge FSyL scenario.

VII. DISCUSSION

Practical Implications and Deployment Considerations: *CODEA* lowers the barrier to deploying federated synergy learning in resource-constrained, multi-edge environments by jointly optimizing slow-timescale container deployment with fast-timescale device-server association. In practice, our framework not only improves training efficiency and model accuracy but also reduces execution cost and update failures. *CODEA* can be integrated with existing container orchestration frameworks, such as Kubernetes or lightweight edge orchestrators, without requiring a fully customized runtime. In such deployments, *CODEA* serves as a control-plane decision module. 1) At the slow timescale, it determines which edge servers should host FSyL containers under the deployment budget. 2) At the fast timescale, it determines device-server associations based on contextual observations and estimated cost benefits. The underlying orchestration system then executes these decisions through container instantiation, service binding, scaling, and, when needed, migration or restart.

The main practical overheads arise from container cold-start delay, image distribution and initialization, and runtime monitoring for deployment updates and association reconfiguration. Importantly, these overheads are incurred primarily when the deployment plan is adjusted, rather than at every local training step, and can therefore be amortized over communication rounds in long-running FSyL tasks. In addition, pre-cached service images or warm standby containers can further reduce deployment latency. Overall, these considerations indicate that *CODEA* is compatible with practical multi-edge systems while leaving room for future system-level optimization.

Limitations and Future Directions: Our current analysis distinguishes between slow-timescale container deployment and fast-timescale edge association, and provides regret bounds under fixed deployment, which leaves a gap in joint, horizon-level performance guarantees for the coupled process. The system model also adopts idealized resource assumptions, including OFDMA-style equal bandwidth slicing and per-device container isolation. While these assumptions are practical and tractable, they may overestimate parallelism in scenarios with bursty

interference. Additionally, translating *CODEA* into a real distributed deployment with hierarchical controllers, control-plane signaling, and fault-tolerant execution remains a challenge. In future work, a unified regret/convergence theory will be developed to jointly cover deployment and association under shared-resource contention, as well as relax equal-share assumptions via measured or learned link scheduling. We will also extend *CODEA* with sub-model extraction and pipeline/model-parallel primitives to support distributed training and serving of large models [45].

VIII. CONCLUSION

This paper addressed the long-term optimization of container deployment and edge association to enable efficient and cost-effective training in multi-edge FSyL systems. We formalized the FSyL benefit maximization problem by jointly modeling deployment and association strategies under resource constraints and system dynamic conditions. To tackle this challenging online optimization problem, we proposed a novel algorithm named *CODEA*, which leverages the contextual multi-armed bandit framework to sequentially optimize container deployment before each global update stage and edge association during each local training round. *CODEA* provides theoretical performance guarantees through a sublinear regret bound, ensuring long-term training optimization without relying on prior knowledge. Extensive experiments demonstrate that *CODEA* consistently outperforms state-of-the-art baselines regarding training efficiency, test accuracy, and resource utilization across diverse FSyL scenarios.

ACKNOWLEDGMENT

The authors also thank the Big Data Computing Center of Southeast University for providing the experiment environment and computing facility.

REFERENCES

- [1] J. Zhang et al., "Edge learning: The enabling technology for distributed Big Data analytics in the edge," *ACM Comput. Surv.*, vol. 54, no. 7, pp. 1–36, 2021.
- [2] R. Laidi et al., "Federated learning in IoT environments: Examining the three-way see-saw for privacy, model-performance, and network-efficiency," *IEEE Commun. Surv. Tut.*, vol. 28, pp. 1025–1058, 2025.
- [3] J. Tu, L. Yang, and J. Cao, "Distributed machine learning in edge computing: Challenges, solutions and future directions," *ACM Comput. Surv.*, vol. 57, no. 5, pp. 1–37, 2025.
- [4] Y. Li, X. Tao, X. Zhang, J. Liu, and J. Xu, "Privacy-preserved federated learning for autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 8423–8434, Jul. 2022.
- [5] X. Xu, H. Tian, X. Zhang, L. Qi, Q. He, and W. Dou, "DisCOV: Distributed COVID-19 detection on X-ray images with edge-cloud collaboration," *IEEE Trans. Serv. Comput.*, vol. 15, no. 3, pp. 1206–1219, May/Jun. 2022.
- [6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
- [7] C. T. Dinh et al., "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 398–409, Feb. 2021.
- [8] Y. Yang, K. Peng, S. Wang, X. Xu, P. Xiao, and V. C. Leung, "Fairness-aware incentive mechanism for multi-server federated learning in edge-enabled wireless networks with differential privacy," *IEEE Trans. Mobile Comput.*, vol. 24, no. 10, pp. 9919–9933, Oct. 2025.

- [9] Y. Cheng et al., "SnapCFL: A pre-clustering-based clustered federated learning framework for data and system heterogeneities," *IEEE Trans. Mobile Comput.*, vol. 24, no. 6, pp. 5214–5228, Jun. 2025.
- [10] S. Fu, F. Dong, D. Shen, J. Zhang, Z. Huang, and Q. He, "Joint optimization of device selection and resource allocation for multiple federations in federated edge learning," *IEEE Trans. Serv. Comput.*, vol. 17, no. 1, pp. 251–262, Jan./Feb. 2024.
- [11] T. Wu et al., "Joint edge aggregation and association for cost-efficient multi-cell federated learning," in *Proc. IEEE Conf. Comput. Commun.*, 2023, pp. 1–10.
- [12] Z. Xu et al., "HierFedML: Aggregator placement and UE assignment for hierarchical federated learning in mobile edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 1, pp. 328–345, Jan. 2023.
- [13] K. Wang, Q. He, F. Chen, H. Jin, and Y. Yang, "FedEdge: Accelerating edge-assisted federated learning," in *Proc. ACM Web Conf.*, 2023, pp. 2895–2904.
- [14] Q. He et al., "Hourglass: Enabling efficient split federated learning with data parallelism," in *Proc. 20th Eur. Conf. Comput. Syst.*, 2025, pp. 1317–1333.
- [15] H. Wang et al., "FedDSE: Distribution-aware sub-model extraction for federated learning over resource-constrained devices," in *Proc. ACM Web Conf.*, 2024, pp. 2902–2913.
- [16] H. Jiang, M. Liu, S. Sun, Y. Wang, and X. Guo, "FedSysL: Computation-efficient federated synergy learning on heterogeneous IoT devices," in *Proc. IEEE/ACM 30th Int. Symp. Qual. Serv.*, 2022, pp. 1–10.
- [17] D. Wu, R. Ullah, P. Harvey, P. Kilpatrick, I. Spence, and B. Varghese, "FedAdapt: Adaptive offloading for IoT devices in federated learning," *IEEE Internet Things J.*, vol. 9, no. 21, pp. 20889–20901, Nov. 2022.
- [18] S. Fu, F. Dong, D. Shen, R. Chen, and J. Hao, "DESIGN: Online device selection and edge association for federated synergy learning-enabled AIoT," *ACM Trans. Intell. Syst. Technol.*, vol. 15, no. 5, pp. 1–28, 2024.
- [19] B. Wu, F. Fang, X. Wang, D. Cai, S. Fu, and Z. Ding, "Client selection and cost-efficient joint optimization for NOMA-enabled hierarchical federated learning," *IEEE Trans. Wireless Commun.*, vol. 23, no. 10, pp. 14289–14303, Oct. 2024.
- [20] Z. Lin et al., "Efficient parallel split learning over resource-constrained wireless edge networks," *IEEE Trans. Mobile Comput.*, vol. 23, no. 10, pp. 9224–9239, Oct. 2024.
- [21] C. Thapa, P. C. M. Arachchige, S. Camtepe, and L. Sun, "SplitFed: When federated learning meets split learning," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 8485–8493.
- [22] W. Wu et al., "Split learning over wireless networks: Parallel design and resource management," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 1051–1066, Apr. 2023.
- [23] E. Samikwa, A. Di Maio, and T. Braun, "ARES: Adaptive resource-aware split learning for Internet of Things," *Comput. Netw.*, vol. 218, 2022, Art. no. 109380.
- [24] Z. Wang, H. Xu, Y. Xu, Z. Jiang, and J. Liu, "CoopFL: Accelerating federated learning with DNN partitioning and offloading in heterogeneous edge computing," *Comput. Netw.*, vol. 220, 2023, Art. no. 109490.
- [25] Z. Cheng et al., "CHEESE: Distributed clustering-based hybrid federated split learning over edge networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 12, pp. 3174–3191, Dec. 2023.
- [26] J. Wu et al., "Fairness-aware budgeted edge server placement for connected autonomous vehicles," *IEEE Trans. Mobile Comput.*, vol. 24, no. 6, pp. 4762–4776, Jun. 2025.
- [27] S. Fu, F. Dong, R. Chen, D. Shen, J. Zhang, and Q. He, "Multi-dimensional training optimization for efficient federated synergy learning," *IEEE Trans. Mobile Comput.*, vol. 24, no. 7, pp. 6243–6258, Jul. 2025.
- [28] S. Yuan et al., "Adaptive incentivize for federated learning with cloud-edge collaboration under multi-level information sharing," *IEEE Trans. Comput.*, vol. 74, no. 7, pp. 2445–2460, Jul. 2025.
- [29] Z. Qu, R. Duan, L. Chen, J. Xu, Z. Lu, and Y. Liu, "Context-aware online client selection for hierarchical federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4353–4367, Dec. 2022.
- [30] K. Aardal, F. A. Chudak, and D. B. Shmoys, "A 3-approximation algorithm for the K-level uncapacitated facility location problem," *Inf. Process. Lett.*, vol. 72, no. 5/6, pp. 161–167, 1999.
- [31] S. Magureanu, R. Combes, and A. Proutiere, "Lipschitz bandits: Regret lower bound and optimal algorithms," in *Proc. Int. Conf. Learn. Theory*, 2014, pp. 975–999.
- [32] N. Wanigasekara and C. L. Yu, "Nonparametric contextual bandits in an unknown metric space," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 14684–14694.
- [33] H. Zhou, D. Feng, Y. Hu, W. Wang, and H. Huang, "Fast garbage collection in erasure-coded storage clusters," *IEEE Trans. Comput.*, vol. 74, no. 8, pp. 2827–2840, Aug. 2025.
- [34] B. Kveton, Z. Wen, A. Ashkan, and C. Szepesvari, "Tight regret bounds for stochastic combinatorial semi-bandits," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2015, pp. 535–543.
- [35] F. Shi, W. Lin, L. Fan, X. Lai, and X. Wang, "Efficient client selection based on contextual combinatorial multi-arm bandits," *IEEE Trans. Wireless Commun.*, vol. 22, no. 8, pp. 5265–5277, Aug. 2023.
- [36] P. Lai et al., "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Proc. 16th Int. Conf. Serv.-Oriented Comput.*, 2018, pp. 230–245.
- [37] F. Chen, J. Zhou, X. Xia, H. Jin, and Q. He, "Optimal application deployment in mobile edge computing environment," in *Proc. IEEE 13th Int. Conf. Cloud Comput.*, 2020, pp. 184–192.
- [38] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C.-H. Hsu, "Edge server placement in mobile edge computing," *J. Parallel Distrib. Comput.*, vol. 127, pp. 160–168, 2019.
- [39] H. Tian, X. Xu, H. Wu, Q. Zhao, J. Dai, and M. Khan, "Cost-efficient deep neural network placement in edge intelligence-enabled Internet of Things," *ACM Trans. Sensor Netw.*, 2024.
- [40] B. Pan, J. Lu, S. Cao, J. Liu, W. Tian, and M. Li, "Bilateral pricing for dynamic association in federated edge learning," *IEEE Trans. Mobile Comput.*, vol. 24, no. 6, pp. 4684–4697, Jun. 2025.
- [41] T. Kim, J. Zuo, X. Zhang, and C. Joe-Wong, "Edge-MSL: Split learning on the mobile edge via multi-armed bandits," in *Proc. IEEE Conf. Comput. Commun.*, 2024, pp. 391–400.
- [42] Z. Zhang, Z. Gao, Y. Guoh, and Y. Gong, "Heterogeneity-aware cooperative federated edge learning with adaptive computation and communication compression," *IEEE Trans. Mobile Comput.*, vol. 24, no. 3, pp. 2073–2084, Mar. 2025.
- [43] Z. Jiang, Y. Xu, H. Xu, Z. Wang, and C. Qian, "Heterogeneity-aware federated learning with adaptive client selection and gradient compression," in *Proc. IEEE Conf. Comput. Commun.*, 2023, pp. 1–10.
- [44] Z. Xu et al., "Energy or accuracy? Near-optimal user selection and aggregator placement for federated learning in MEC," *IEEE Trans. Mobile Comput.*, vol. 23, no. 3, pp. 2470–2485, Mar. 2024.
- [45] Y. Cheng, W. Zhang, Z. Zhang, C. Zhang, S. Wang, and S. Mao, "Toward federated large language models: Motivations, methods, and future directions," *IEEE Commun. Surv. Tut.*, vol. 27, no. 4, pp. 2733–2764, Aug. 2025.
- [46] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," 2018, *arXiv:1812.00564*.
- [47] Y. Gao et al., "End-to-end evaluation of federated learning and split learning for Internet of Things," in *Proc. Int. Symp. Reliable Distrib. Syst.*, 2020, pp. 91–100.
- [48] R. Deng, X. Du, Z. Lu, Q. Duan, S.-C. Huang, and J. Wu, "HSFL: Efficient and privacy-preserving offloading for split and federated learning in IoT services," in *Proc. 2023 IEEE Int. Conf. Web Serv.*, 2023, pp. 658–668.
- [49] B. Huang, H. Zhao, L. Wang, W. Qian, Y. Yin, and S. Deng, "Decentralized proactive model offloading and resource allocation for split and federated learning," *IEEE Internet Things J.*, vol. 12, no. 9, pp. 12226–12241, May 2025.
- [50] L. U. Khan, M. Guizani, A. Al-Fuqaha, C. S. Hong, D. Niyato, and Z. Han, "A joint communication and learning framework for hierarchical split federated learning," *IEEE Internet Things J.*, vol. 11, no. 1, pp. 268–282, Jan. 2024.
- [51] J. Tirana, D. Tsigkari, G. Iosifidis, and D. Chatzopoulos, "Workflow optimization for parallel split learning," in *Proc. IEEE Conf. Comput. Commun.*, 2024, pp. 1331–1340.
- [52] Z. Lin et al., "Hierarchical split federated learning: Convergence analysis and system optimization," *IEEE Trans. Mobile Comput.*, vol. 24, no. 10, pp. 9352–9367, Oct. 2025.



Shucun Fu received the PhD degree in computer science and technology from Southeast University, in 2025. Currently, he is a lecturer with the School of Computer Science, School of Software, Nanjing University of Information Science and Technology, China. His research interests include edge intelligence, federated learning, and service computing. He serves as the chair of ACM Nanjing Student Chapter.



Fang Dong (Member, IEEE) received the BS and MS degrees in computer science from the Nanjing University of Science & Technology, China, in 2004 and 2006, respectively, and the PhD degree in computer science from Southeast University, in 2011. He is currently a professor with the School of Computer Science and Engineering, Southeast University, China. His current research interests include edge intelligence, cloud computing, and Industrial Internet. He is a member of the ACM, he also served as the co-chair of ACM Nanjing Chapter.



Xiaolong Xu (Senior Member, IEEE) received the PhD degree in computer science and technology from Nanjing University, China, in 2016. He is currently a full professor with the School of Computer Science, School of Software, Nanjing University of Information Science and Technology. He has published more than 100 peer-review articles in international journals and conferences, including the *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE/ACM*

Transactions on Networking, *IEEE Transactions on Services Computing*, *IEEE Transactions on Mobile Computing*, *ACM Transactions on Information Systems*, *SCIS*, *ICML*, *NeurIPS*, *WWW*, *ACM SIGIR*, *CVPR*, *IJCAI*, *AAAI*, *ACM MM*, etc. He was selected as the highly cited researcher of Clarivate (2021–2024). He received best paper awards from the *Tsinghua Science and Technology* at 2023, *Journal of Network and Computer Applications* at 2022, and several conferences, including IEEE HPCC 2023, IEEE ISPA 2022, IEEE CyberSciTech 2021, IEEE CPSCOM2020, etc. His research interests include large-small model collaboration, edge intelligence, recommender systems, service computing, and Big Data.



Dian Shen received the bachelor's, master's, and PhD degrees from Southeast University, China, in 2010, 2012, and 2018, respectively. He is an associate professor with the School of Computer Science and Engineering, Southeast University, China. He has been a visiting researcher with Coventry University, U.K. and the Chinese University of Hong Kong, in 2013 and 2021, respectively. His research interests include edge computing, virtualization, and data center network.



Muhammad Bilal (Senior Member, IEEE) received the PhD degree in information and communication network engineering from the School of Electronics and Telecommunications Research Institute (ETRI), Korea University of Science and Technology, in 2017. From 2017 to 2018, he was with Korea University, where he was a postdoctoral research fellow with the Smart Quantum Communication Center. In 2018, he joined Hankuk University of Foreign Studies, South Korea, where he was an associate professor with the Division of Computer and Electronic Systems Engineering. In 2023, he joined Lancaster University as a senior lecturer (associate professor) with the School of Computing and Communications. He has led and contributed to funded projects from EU Horizon, IITP, MOTIE, BK21, NRF Korea, and NSFC China. He is a prolific author and holds multiple US and Korean patents. He serves on editorial boards for journals including the *IEEE Transactions on Intelligent Transport Systems*, *IEEE Internet of Things Journal*, *Alexandria Engineering Journal*, and *Physical Communication*. He is also co-editor-in-chief of the *International Journal of Smart Vehicles and Smart Transportation*. Additionally, he contributes to major conferences such as IEEE VTC, ICC, CCNC, and ACM SigComm as a TPC member.



Schahram Dustdar (Fellow, IEEE) is a full professor of computer science and heads TU Wien's Distributed Systems Group. He is also part time ICREA research professor with ICREA, Barcelona, Spain. His research interests include distributed systems, edge intelligence, complex, and autonomic software systems. He's the editor in chief of the *Computing*; associate editor of the *ACM Transactions on the Web*, *ACM Transactions on Internet Technology*, *IEEE Transactions on Cloud Computing*, and *IEEE Transactions on Services Computing*. He's also on the editorial boards of the *IEEE Internet Computing* and *IEEE Computer*. He has received the ACM Distinguished Scientist award and Distinguished Speaker Award and the IBM Faculty Award. He is an elected member of Academia Europaea, where he's was Informatics Section chairperson from 2015 to 2022. He is a fellow of the AAIA and AIIA.