Optimal Resource Allocation for AIoT as a Service Under Various Service Scenarios and Architectures

Ren-Hung Hwang, *Senior Member, IEEE*, Tsai-Ying Chou, Jia-You Lin, *Student Member, IEEE*, Didik Sudyana, Yuan-Cheng Lai, and Ying-Dar Lin, *Fellow, IEEE*

***Abstract*—The integration of artificial intelligence (AI) with the Internet of Things (IoT) marks a significant advancement in sixth-generation (6G) networks. The complexity of these AIoT services has promoted an as-a-service model, where service providers offer tailored architectures to meet varied application needs. Despite the critical importance of optimizing both training and inference in service architectures, this aspect remains under-explored. Our study introduces service scenarios such as 'no shared (NS)', where tenants manage their data and models independently, 'data shared (DS)', where tenants provide data for collective training, and 'parameter sharing (PS)', where only model parameters are shared. We utilize a tandem queue model to simulate the communication and computing demands across cloud-edge-fog architectures. Our proposed Cost and Delay Resource Allocation (CDRA) method significantly reduces costs, with edge and fog-based training and inference lowering costs by up to 44% compared to cloud setups. The evaluation shows that the NS scenario is resource-intensive but offers high privacy, DS is cost-effective and improves model accuracy, and PS balances privacy with longer wait times. These findings provide service providers with a comprehensive comparison of service scenarios and architectures, offering guidance for strategic and economically sound decisions in the ever-evolving landscape of AIoT.**

***Index Terms*—AIoT as a service, resource optimization, distributed learning, service scenario, service architecture.**

I. Introduction

THE ARTIFICIAL Intelligence of Things (AIoT) is set to be a cornerstone of the upcoming sixth-generation (6G) network [1], [2]. Integrating AI with IoT offers advantages such as improved efficiency and smarter data interpretation. However, the complexities of developing and managing these systems present challenges, necessitating specialized knowledge. Many enterprises find it costly and time-intensive to onboard such expertise, leading them to outsource their AIoT management to service providers. This outsourcing trend marks a shift towards AIoT being offered as a service, which allows the enterprises, acting as tenants, to benefit from specialized knowledge and resources without the burdens of in-house management.

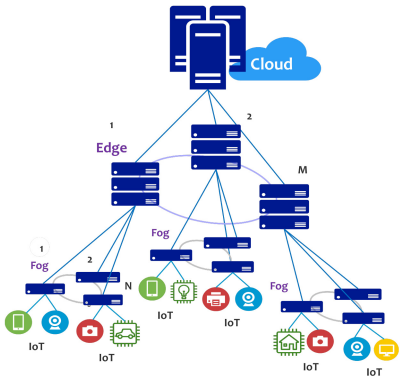


Fig. 1. Multi-tier architecture for AIoT system.

Service providers must carefully choose the service architecture for offering AIoT as a Service (AIoTaS) platforms [3]. Several computing paradigms, including cloud, edge, and fog computing, support AIoT services through a multi-tier architecture. Figure 1 shows this architecture, based on frameworks by the European Telecommunication Standards Institute (ETSI) [4] and the 5G-CORAL architecture [5]. The cloud tier provides scalability, resource pooling, cost efficiency, and ease of maintenance, managed by providers such as Google and Amazon. The edge tier, initiated by ETSI, virtualizes cloud capabilities within mobile networks and includes operators such as T-Mobile and Verizon. The fog tier extends cloud and edge capabilities closer to data sources, supporting real-time data processing on local devices [6]. This hierarchical structure allows service providers to manage data for various AI training modes, resulting in diverse service architectures and improving model comprehensibility in real-world scenarios.

Regarding tenant data management, service providers have several options of service scenarios. Tenants can independently manage their data and models without sharing information with other tenants, known as 'no shared (NS).' Alternatively, tenants may provide their data to service providers, who aggregate it as the training data through distributed learning, termed 'data shared (DS).' Lastly, tenants can share only

Received 25 November 2023; revised 24 June 2024; accepted 16 November 2024. Date of publication 20 November 2024; date of current version 22 April 2025. The associate editor coordinating the review of this article and approving it for publication was J.-H. Cho. *(Corresponding author: Didik Sudyana.)*

Ren-Hung Hwang is with the College of Artificial Intelligence, National Yang Ming Chiao Tung University, Tainan 71150, Taiwan (e-mail: rhhwang@nycu.edu.tw).

Tsai-Ying Chou and Jia You Lin are with the Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi 621301, Taiwan.

Didik Sudyana and Ying-Dar Lin are with the Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu 30010, Taiwan (e-mail: dsudyana@cs.nycu.edu.tw; ydlin@cs.nycu.edu.tw).

Yuan-Cheng Lai is with the Department of Information Management, National Taiwan University of Science and Technology, Taipei 106, Taiwan (e-mail: laiyc@cs.ntust.edu.tw).

Digital Object Identifier 10.1109/TNSM.2024.3503575

model parameters, enabling service providers to aggregate them through distributed learning, called 'parameter shared' (PS), which includes federated learning (FL) and decentralized learning (DL). These distributed methods facilitate parameter exchange without exposing raw data, thus maintaining privacy.

Service providers must understand which AIoT service configurations can meet tenant demands while minimizing operational costs. Cost management is crucial for maximizing revenue. By analyzing service architectures, providers can determine optimal configurations for tenants.

Previous research highlights the benefits of edge computing for AI services, offering faster and more reliable data processing by utilizing physical edge nodes [7], [8], [9], [10], [11], [12], [13], [14], [15], [16]. However, these studies mainly focus on resource optimization within single service scenarios, overlooking the analysis of service and network architectures from a service provider's perspective. Our research aims to analyze trade-offs between cost, data privacy, delay, accuracy, and collaboration within AIoTaS, providing recommendations that address service trade-offs not extensively covered in previous studies.

In this study, we model the service architectures within three different service scenarios, i.e., NS, DS, and PS, and develop a delay evaluation model for both the training and inference processes. Subsequently, we introduce Cost and Delay Resource Allocation (CDRA) optimization algorithm with the goal to minimize the cost. For each service architecture, we set the arrival rate and service demand parameters of all communication and computation queues to derive the average training delay and average inference delay of an arrival task through queueing theory. We also set the delay constraint of training and inference individually according to the application scenario. Under such constraint, resources must be allocated to each queue in order to get the minimum total resource cost.

Furthermore, to deepen our understanding, we examine different service architectures and scenarios, focusing on three key areas: (1) evaluating resource allocation algorithms by comparing our Cost and Delay Resource Allocation (CDRA) method with simulated annealing; (2) analyzing the NS scenario to understand how various training modes and inference positions affect resource costs, server costs, and waiting times, including the impact of raw data sizes and batch sizes; and (3) assessing the performance across all service scenarios.

Our work's contributions to the field are threefold:

- *Comprehensive Analysis:* We provide a detailed examination of service scenarios and architectures, highlighting critical trade-offs between cost, data privacy, delay, accuracy, and collaboration. This enables service providers to make informed decisions about their AIoTaS deployments, addressing the literature gap regarding the integration of multiple service perspectives and network architectures.
- *Optimization Model:* We introduce a new optimization model to minimize resource costs while meeting delay constraints in training and inference. The model utilizes CDRA algorithm for efficiency, complemented by Simulated Annealing for achieving near-optimal solutions quickly and effectively.
- *Holistic Insights:* Our study explores the intricate dynamics of various service architectures and scenarios, providing a broad view that aids service providers in understanding and managing the complexities associated with AIoTaS configurations.

The rest of this paper is structured as follows: Section II reviews relevant literature. Section III introduces our queueing model for service architectures and formulates the optimization problem to minimize resource costs. In Section IV, we describe our CDRA algorithm designed to solve this problem. Section V discusses simulation results and their analysis. The paper concludes with Section VI.

II. Related Work

AIoT architecture research has been thoroughly explored with diverse optimization objectives, including delay reduction and cost efficiency. Table I provides a summary of related research, detailing the optimization goals, utilized architectures, number of services addressed, focused tasks, and optimization algorithms employed by various researchers.

Starting from the group of minimizing delay in the edge architecture, the work of [7] employed swarm intelligence to optimize resource allocation with the intent of minimizing both delay and cost. Similarly, [8] optimized task scheduling for the AIoT system. Going further in this category, [9] proposed the IoT-SCOM algorithm to optimize data transfer latency to derive an ideal transfer deployment strategy. Reference [10] explored the potential of the BTO algorithm to optimize offloading in AIoT-enabled MEC networks.

In the realm of cost optimization, several works have been presented. For instance, [11] utilized DQN (Deep Q-Network) for the optimization of resource allocation aiming to maximize system revenue. Reference [12] emphasized a joint optimization strategy for computation, caching, and communication in the AIoT system. Reference [13] investigated the trade-offs between inference accuracy, latency, and resource cost. Furthermore, using convex optimization solvers, [14] pursued minimizing time-averaged cost within set energy budget and delay constraints. The works from [15] and [16] also contributed to this category, probing the relationship between AIoT application performance and energy cost, and striving to minimize the composite of system cost and learning cost, respectively.

Our research significantly advances beyond the typical focus of previous studies on resource optimization within isolated service scenarios, which often overlook broader analyses crucial for AIoT as a Service (AIoTaS) implementations. Key advancements of our work include:

- *Holistic Service Architecture Analysis:* We explore a range of service architectures, from simple single-tier systems to complex multi-tier configurations, extending beyond the common cloud-edge setups.
- *Balanced Optimization Focus:* Our methodology integrates multiple factors—cost, data privacy, accuracy, and collaboration—into a unified analytical framework, providing a deeper understanding of their interplay and impact.

TABLE I
A COMPARISON OF RELATED WORKS ON RESOURCE ALLOCATION OPTIMIZATION FOR AIOT

Paper	Obj	Arch	# of services	AI Task		CL	DL/FL		Resource Opt		Optimization Algorithm	Purpose
				Tr	In		1-tier	2-tier	Comm	Comp		
[7]	Min delay	C-E	1	X	O	X	X	X	X	O	Swarm Intelligence	Optimizing resource allocation to minimize delay and cost
[8]		C-E							X	O		Optimizing task scheduling for AIoT system
[9]		C-E							O	X	IoT-SCOM	Optimizing the data transfer latency to find the ideal transfer deployment strategy
[10]		UAV-MEC							X	O	BTO algorithm	Optimizing the offloading in AIoT-enabled MEC networks
[11]	Min Cost	C-E							X	O	DQN	Optimizing resource allocation to maximize system revenue
[12]		C-E							O	O	Smart-Edge-CoCaCo	Joint optimization of computation,caching, and communication in AIoT system
[13]		C-E							X	O	MOE/D	Investigating the relationship of the AIoT application performance and the energy cost
[14]		E		O	X		O		O	X	Convex Optimization Solvers	Minimizing the time-averaged cost under the given long-term energy budget and delay constraints
[15]		E			O	O	X		X	O	Join Optimization Framework	Investigating the three-way trade-off between inference accuracy, latency,and resource cost
[16]		E				X	O		O	O	Join Resource Optimization	Minimizing the weighted sum of system cost and learning cost
Ours		C-E-F	3			O	O	O	O	O	CDRA	Providing recommendations to service providers regarding service scenarios and network architectures of AIoTaS based on optimization results

C/E/F: Cloud/Edge/Fog Tr/In: Training/Inference CL/DL/FL: Centralized Learning/Distributed Learning/Federated Learning
Obj/Arch: Objective/Architecture Comm/Comp: Communication/Computation

- *Comprehensive Recommendations:* We provide actionable recommendations for service providers, optimizing AIoTaS configurations considering technical, operational, and strategic factors.
- *In-depth Examination of Service Scenarios:* Our detailed exploration of various service scenarios equips providers with a thorough understanding of the complexities and strategic implications of different AIoTaS configurations.

By covering these areas comprehensively, our study offers a holistic perspective that aids service providers in effectively deploying and managing AIoTaS, bridging a significant gap in the existing literature and enhancing the practical relevance of our findings.

III. SYSTEM MODEL AND PROBLEM STATEMENT

This section describes the three-tier network architecture, the service architectures in three service scenarios, their queueing models (i.e., the delay analysis models), and problem definition. The key notations used in this paper are summarized in Table II.

A. System Architecture and Service Scenarios

In the AIoTaS system, there are K tenants and a three-tier network system architecture, as shown in Fig. 1. K tenants jointly use a cloud server associated with M edge servers and N fog servers which the edge responsible for receiving messages from IoT devices. Additionally, network slicing is used to provide each tenant with their own communication and computing resources. We then defined these resources as W^x for $x \in \mathcal{I}$, where $\mathcal{I}$ represents the set of computation nodes and communication links, defined as $\mathcal{I} = \mathcal{I}^{\mathcal{N}} \cup \mathcal{I}^{\mathcal{S}}$. Here, $\mathcal{I}^{\mathcal{N}}$ and $\mathcal{I}^{\mathcal{S}}$ are the sets of communication links and computation nodes, respectively, with $\mathcal{I}^{\mathcal{N}} \in \{\mathrm{FE}, \mathrm{EC}, \mathrm{CE}, \mathrm{EF}, \mathrm{FF}, \mathrm{EE}\}$ and $\mathcal{I}^{\mathcal{S}} \in \{\mathrm{F}, \mathrm{E}, \mathrm{C}\}$. The symbols F, E, and C denote fog, edge, and cloud nodes, respectively.

As for model training in AIoTaS, it is essential to consider whether the data collected by each tenant should be shared or not. Based on the sharing mode, tenants can adopt three service scenarios for different applications. Below is a detailed description of the service scenarios and their features.

1) No Shared (NS): In this service scenario, each tenant trains their own model using their local data to protect data privacy. The data is completely independent and not shared with other tenants. For example, a hospital that sources its training dataset from its medical records can implement smart healthcare services while keeping patient data secure.

2) Data Shared (DS): "Data shared" refers to a scenario where all tenants' data is aggregated to train a global model. This approach grants tenants access to a more knowledgeable and accurate model. However, it comes with the trade-off of exposing their data to the public. The motivation behind this service is to harness the collective intelligence of a tenant community, offering access to a high-quality model capable of superior performance and generalization.

3) Parameter Shared (PS): To maintain privacy and facilitate collaborative model training, sharing only the final ML model parameters instead of the original data may be preferred. In this scenario, each tenant independently trains its ML model, and the central server aggregates all model parameters through FL or DL to derive a global model. This approach enables private data to be kept confidential while allowing for collaboration and developing a more accurate model.

B. Various Hierarchical Service Architectures

When considering service architectures, it is essential to account for model training and the location of model inference. This paper examines two model training mechanisms: CL and multi-tier FL or DL [17]. In CL, data is centrally trained in the cloud, while in multi-tier learning, lower-level servers handle

TABLE II
SUMMARY OF KEY NOTATIONS

Notation	Meaning
Topology	
K	The number of tenants
M	The number of edge servers for each tenant
N	The number of fog servers under each edge server
$\mathcal{I}$	The set of computation nodes and communication links, $\mathcal{I} \in \mathcal{I}^{\mathcal{N}} \cup \mathcal{I}^{\mathcal{S}}$
$\mathcal{I}^{\mathcal{N}}$, $\mathcal{I}^{\mathcal{S}}$	The set of communication links and computation nodes, $\mathcal{I}^{\mathcal{N}} \in$ {FE, EC, CE, EF, FF, EE}, $\mathcal{I}^{\mathcal{S}} \in$ {F, E, C}
Num^x	The number of nodes, $Num^x \in \{MN, M, M, MN, \frac{N(N-1)}{2}, \frac{M(M-1)}{2}, MN, M, 1\}$, $x \in \mathcal{I}$
Arrival Rate and Service Demand	
λ	Arrival rate of raw data collected by each tenant
μ_1^L, μ_2^L	Average size of each arrival raw data and model parameters
μ_1^T, μ_1^P	Average ML model training and inference workload
μ_1^F, μ_2^F	Average DL and FL workload
μ_3^F	Each tenant's local model aggregates into a global model, average FL workload for each tenant
$\mathcal{S}_{i,x,l}^{T,m}$, $\mathcal{A}_{i,x,l}^{T,m}$	Service demand and arrival rate of node x in m service scenario's i-th service architectures during the training process, $x \in \mathcal{I}$. $l \in \{1,2\}$ is used to differentiate between two arrival tasks.
$\mathcal{S}_{k,x}^{P,m}$, $\mathcal{A}_{k,x}^{P,m}$	Service demand and arrival rate of node x in m service scenario's k-th inference situations during inference process, $x \in \mathcal{I}$.
Delay	
$D_i^{T,m}$, $D_i^{P,m}$	Training and inference delay in m service scenario's i-th service architectures
$D_{i,x}^{T,m}$, $D_{i,x}^{P,m}$	Queue delay for each queue in m service scenario's i-th service architectures during training and inference process, $x \in \mathcal{I}$.
Z^x	Propagation delay and $x \in \mathcal{I}^{\mathcal{N}}$.
δ^T, δ^P	Training and inference delay constraint
$D_{i,t}^{W,m}$, $D_{i,f1}^{W,m}$, $D_{i,f2}^{W,m}$, $D_{i,f3}^{W,m}$	The waiting time for one data point on the training server, first aggregation server, second aggregation server, and K-tenant aggregation server, respectively.
Model Training Rate	
B_1	In average, after training B_1 samples of raw data, the first aggregation will be triggered.
B_2	In average, after B_2 rounds of first aggregation, second aggregation will be evoked.
B_3	In average, after B_3 second aggregations, K-tenant aggregation will be evoked
B_4	In average, after B_4 rounds of K-aggregation, final model will be sent back to the training server.
Resource and Resource Cost	
$R_{i,x}^{P,m}$, $R_{i,x}^{T,m}$	During inference and training process, the allocated resource to each queue x in m service scenario's i-th service architectures, x $\in \mathcal{I}$.
R_{unit}	Unit allocated resource
W^x	The resource cost of computaion nodes and communication links, x $\in \mathcal{I}$.
$C_i^{T,m}$,$C_i^{P,m}$, $\mathbf{C}_i^m$	Training resource cost, Inference resource cost, and total resource cost in m service scenario's i-th service architectures

training, and upper-level servers use FL or DL to aggregate model parameters.

FL and DL differ based on parameter aggregation layers, influenced by the presence or absence of a parameter server. We categorize layers as fog (Layer 3), edge (Layer 2), and Cloud (Layer 1). FL requires aggregation at a higher layer $(y < x)$ than the training layer (x), while DL aggregates parameters at the same layer (x). If FL follows DL at the same layer, then the model aggregation is performed twice, and the FL server should be at layer $x-2$. For instance, if the model trains at the fog layer first, and aggregates models using DL, then FL cannot occur at the edge; it needs the cloud level to reaggregate model parameters.

In the fog-edge-cloud network architecture, combining FL and DL enables hierarchical learning during model training. There are several hierarchical learning architectures based on where FL or DL occurs—fog, edge, or cloud levels.

Parameter aggregation in AIoTaS architecture varies by scenario. Typically, one-tier aggregation occurs at the cloud level post-training at edge or fog. Two-tier aggregation involves: FL at the edge then at the cloud, a mix of FL and DL at the edge, and DL at fog nodes followed by FL at the cloud. In *K*-tenant aggregation, each tenant aggregates models using FL, requiring an extra step for global model synthesis. In all cases, the original training host must receive the final results to update parameters for subsequent training rounds.

Table III provides an overview of training modules for NS, DS, and PS service scenarios, analyzing server-level model training, aggregation locations, and the number of models generated. Notations follow the format of ST-A1-A2+A3, where S represents the service scenario (NS: N, DS: D, PS: P), T represents the training location, A1 and A2 denote the first and second parameter aggregation locations (FL or DL), and A3 represents *K*-tenants aggregation. In terms of model numbers, higher-level hosts aggregate lower-level host models to obtain a shared model. For example, in NF-E-C training mode, each tenant trains their model in the fog, resulting in $M \times N \times K$ models. The edge then performs the first parameter aggregation from the N bottom fogs, resulting in $M \times K$ aggregated models. Finally, the cloud aggregates parameters from Edge to obtain K public models for K tenants. In the DS scenario, K tenants combine data to train one global model, while in the PS scenario, a single global model is achieved through *K*-tenant aggregation.

After training, determining the inference location is also crucial in the AIoTaS system. Each service architecture is represented as ST-A1-A2+A3/*I*, with *I* indicating the server for model inference.

C. Queueing Model of Various Hierarchical Training Architecture in Three Service Scenarios

We define a queuing model for training architectures across three service scenarios within a homogeneous network. This model views the Fog-Edge-Cloud sequence as a linear queue for communication and computation. The NF-E-C/F architecture exemplifies this, with associated diagrams and queuing models depicted in Figures 2 and 3. These illustrations clarify the nature of data transfers (whether raw data or model parameters) and server tasks such as training, parameter aggregation, and inference, emphasizing variations in data volumes and resource demands.

Specifically, we use λ to denote the arrival rate of tenant-collected raw data. In training, a Fog node possesses an arrival rate of λ/MN (with *MN* representing the Fog node total). Upon data arrival, the model adjusts its parameters, transmitting model parameters to the connected Edge node after training B_1 raw data sets, where B_1 is the expected batch

TABLE III
VARIOUS TRAINING ARCHITECTURES IN THREE SERVICE SCENARIOS

Service Scenarios	Centralized		One-tier aggregation		Two-tier aggregation	
	Training Mode	# of models	Training Mode	# of models	Training Mode	# of models
No Shared	NC	K	NE-C	$MK \rightarrow K$	NF-E-C	$MNK \rightarrow MK \rightarrow K$
			NF-C	$MNK \rightarrow K$	NF-E-E	$MNK \rightarrow MK \rightarrow K$
					NF-F-C	$MNK \rightarrow MK \rightarrow K$
Data Shared	DC	1	DE-C	$M \rightarrow 1$	DF-E-C	$MN \rightarrow M \rightarrow 1$
			DF-C	$MN \rightarrow 1$	DF-E-E	$MN \rightarrow M \rightarrow 1$
					DF-F-C	$MN \rightarrow M \rightarrow 1$
Parameter Shared	Centralized + K-tenant aggregation		One-tier + K-tenant aggregation		Two-tier + K-tenant aggregation	
	Training Mode	# of models	Training Mode	# of models	Training Mode	# of models
	PC+C	$K \rightarrow 1$	PE-C+C	$MK \rightarrow K \rightarrow 1$	PF-E-C+C	$MNK \rightarrow MK \rightarrow K \rightarrow 1$
			PF-C+C	$MNK \rightarrow K \rightarrow 1$	PF-E-E+C	$MNK \rightarrow MK \rightarrow K \rightarrow 1$
					PF-F-C+C	$MNK \rightarrow MK \rightarrow K \rightarrow 1$

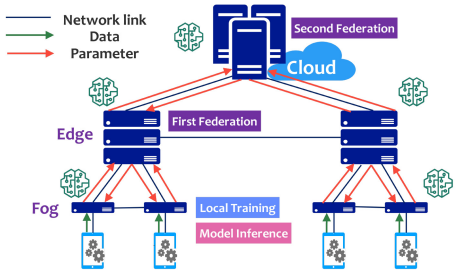


Fig. 2. Flow of data and model parameters and model training, model parameter aggregation, and model inference position in NF-E-C/F.

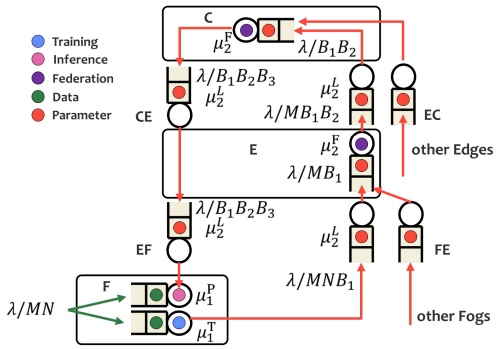


Fig. 3. The queueing model of NF-E-C/F.

size. The batch size is a random variable with B_1 as its mean to maintain the Poisson arrival property. The FE queue's arrival rate is λ/MNB_1, while the Edge node, which consolidates parameters from Fog nodes, has λ/MB_1.

The choice to model arrival processes using a Poisson distribution is based on its widespread application in scenarios where events occur independently over a period of time, which is a common characteristic in network traffic and IoT environments. In the context of our AIoT system, the arrival of data packets or tasks to different architecture components (cloud, edge, and fog layers) can be assumed to be random and independent from one another. This assumption is supported by the nature of IoT devices and applications, which typically generate data asynchronously and at varying intervals. For example, sensors in a smart city environment may send data based on event triggers (e.g., traffic movement or environmental changes), which naturally align with the stochastic model provided by the Poisson distribution.

At the edge, training parameters merge with the edge model's existing parameters. After B_2 initial federation rounds, these are sent to the cloud for a second round. This model aggregation results in the EC queue having λ/MB_1B_2 and the cloud node accumulating at λ/B_1B_2. After B_3 secondary aggregation rounds, the final cloud model returns to the training server, giving the CE and EF queues an arrival rate of $\lambda/B_1B_2B_3$. For inference, the fog receives raw data at a rate of λ/MN.

From Fig. 3, it is evident that communication demand and computational overhead influence each queue's resource needs. Specifically, communication demand refers to data transmission size, with μ_1^L and μ_2^L indicating the average size of raw data and ML model parameters. Computational overhead includes average workloads for ML model training, inference, DL, and FL, symbolized by μ_1^T, μ_1^P, μ_1^F, and μ_2^F.

Using the NF-E-C/F model, we determine service demand and arrival rate parameters across all architectures, as detailed in Tables IV and V for NS scenarios in inference and training. Inference can occur at Cloud, Edge, or Fog levels. For Cloud-based inference, raw data flows through FE and EC queues at an arrival rate of λ. For Edge or Fog-based inference, the arrival rate is reduced to λ/M and λ/MN, respectively, due to distributed data collection. The communication overhead remains constant at μ_1^L, and computation demand at μ_1^P.

The training process is influenced by the location of inference servers, especially in terms of communication delays for transmitting model parameters. For instance, in the NC scenario, after B_1 training rounds, parameter data is transmitted to Edge or Fog servers at a rate of λ/B_1. Cloud-based inference avoids this delay as parameters are already located in the Cloud. Table V focuses on cloud-based inference training, with alternative configurations for Edge and Fog-based inference detailed similarly and included in the Appendix.

D. Delay Model

In this subsection, we determine average computation node and communication link delays with maximum slicing resources for inference and training processes, based on Tables IV and V. The following provides a detailed description of the average inference delay, average training delay, and waiting time.

TABLE IV
THE ARRIVAL RATE AND SERVICE DEMAND PARAMETERS WITH DIFFERENT INFERENCE SERVER

Inference server	$\mathcal{A}/\mathcal{S}$	ID	Communication (data size)						Computation (workload)		
			FE	EC	CE	EF	FF	EE	F	E	C
C	$\mathcal{A}$	k=1	λ/MN	λ/M	0	0	0	0	0	0	λ
	$\mathcal{S}$		μ_1^L	μ_1^L	0	0	0	0	0	0	μ_1^P
E	$\mathcal{A}$	k=2	λ/MN	0	0	0	0	0	0	λ/M	0
	$\mathcal{S}$		μ_1^L	0	0	0	0	0	0	μ_1^P	0
F	$\mathcal{A}$	k=3	0	0	0	0	0	0	λ/MN	0	0
	$\mathcal{S}$		0	0	0	0	0	0	μ_1^P	0	0

TABLE V
THE ARRIVAL RATE AND SERVICE DEMAND PARAMETERS IN NS DURING TRAINING PROCESS

Service Architectures	$\mathcal{A}/\mathcal{S}$	**ID**	**Communication (data size)**						**Computation (workload)**		
			FE	EC	CE	EF	FF	EE	F	E	C
NC/C	$\mathcal{A}$	i=1	λ/MN	λ/M	0	0	0	0	0	0	λ
	$\mathcal{S}$		μ_1^L	μ_1^L	0	0	0	0	0	0	μ_1^T
NE-C/C	$\mathcal{A}$	i=4	λ/MN	λ/MB_1	λ/B_1B_2	0	0	0	0	λ/M	λ/B_1
	$\mathcal{S}$		μ_1^L	μ_2^L	μ_2^L	0	0	0	0	μ_1^T	μ_2^F
NF-C/C	$\mathcal{A}$	i=7	λ/MNB_1	λ/MB_1	λ/B_1B_2	λ/B_1B_2	0	0	λ/MN	0	λ/MN
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	0	μ_1^T	0	μ_2^F
NF-E-C/C	$\mathcal{A}$	i=10	λ/MNB_1	λ/MB_1B_2	$\lambda/B_1B_2B_3$	$\lambda/B_1B_2B_3$	0	0	λ/MN	λ/MB_1	λ/B_1B_2
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	0	μ_1^T	μ_2^F	μ_2^F
NF-E-E/C	$\mathcal{A}$	i=13	λ/MNB_1	$\lambda/B_1B_2B_3$	0	$\lambda/B_1B_2B_3$	0	λ/MB_1B_2	λ/MN	λ/MB_1, λ/B_1B_2	0
	$\mathcal{S}$		μ_2^L	μ_2^L	0	μ_2^L	0	μ_2^L	μ_1^T	μ_2^F, μ_1^F	0
NF-F-C/C	$\mathcal{A}$	i=16	λ/MB_1B_2	λ/MB_1B_2	$\lambda/B_1B_2B_3$	$\lambda/B_1B_2B_3$	λ/MNB_1	0	λ/MN, λ/MB_1,	0	λ/B_1B_2
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	μ_1^T, μ_1^F	0	μ_2^F

1) Average Inference Delay: The arrival process of each queue follows Poisson distribution, and the service time follows an exponential distribution. Therefore, we can utilize the M/M/1 queuing model to obtain the delay. According to Little's law in queuing theory, the average delay for each queue under *i*-th service architecture can be calculated as

$$D_{i,x}^{P,m} = \begin{cases} \frac{1}{\frac{R_{i,x}^{P,m}}{\mathcal{A}_{k,x}^{P,m}} - \mathcal{S}_{k,x}^{P,m}}, & \text{if } \mathcal{A}_{k,x}^{P,m} > 0 \\ 0, & \text{otherwise.} \end{cases}, \tag{1}$$

where $m \in \{\text{NS, DS, PS}\}$ and $R_{i,x}^{P,m}$ represents the resources allocated to the *x* queue through network slicing. In Table IV, $k = 1, 2, 3$. $\mathcal{A}_{k,x}^{P,m}$ and $\mathcal{S}_{k,x}^{P,m}$ represent the arrival rate and service demand of *x* queue individually. Due to queue delay $D_{i,x}^{P,m} > 0$, we can set the lower bound $R_{i,x}^{P,m}$ as

$$LB\_R_{i,x}^{P,m} = \begin{cases} \mathcal{A}_{k,x}^{P,m}\mathcal{S}_{k,x}^{P,m}, & \text{if } \mathcal{A}_{k,x}^{P,m} > 0 \\ 0, & \text{otherwise.} \end{cases}. \tag{2}$$

Therefore, we can derive the average delay required for one data point in the inference process as

$$D_i^{P,m} = \sum_{x \in \mathcal{I}} D_{i,x}^{P,m} + \sum_{n \in \mathcal{I}^{\mathcal{N}}} Z^n, \tag{3}$$

$$Z^n = \frac{D^n}{\frac{1}{RI} \times LS}, \tag{4}$$

where Z^n is the propagation delay of a network link *n* between two servers; and where D^n is the distance of a network link *n*, *RI* represents the refractive index of fiber optic, and *LS* is the speed of light, approximately $3 \times 10^8 m/s$

2) Average Training Delay: For the training process, model training and parameter aggregation may be performed on the same server. Therefore, when calculating the average queue delay, we utilize both the M/M/1 and M/H_2/1 queueing models. Specifically, the M/M/1 model is used when there is only one type of arrival tasks in the queue, while the M/H_2/1 model is adopted when there are two types of arrival tasks. Consequently, the delay of each queue in service architecture *i* during the training process can be calculated as

$$D_{i,x}^{T,m} = \begin{cases} \frac{1}{\frac{R_{i,x}^{T,m}}{\mathcal{A}_{i,x,1}^{T,m}} - \mathcal{S}_{i,x,1}^{T,m}}, & \text{if } \mathcal{A}_{i,x,1}^{T,m} > 0, \mathcal{A}_{i,x,2}^{T,m} \le 0 \\ \mathbb{E}[S] + \frac{\rho \mathbb{E}[S^2]}{2(1-\rho)\mathbb{E}[S]}, & \text{if } \mathcal{A}_{i,x,1}^{T,m} > 0, \mathcal{A}_{i,x,2}^{T,m} > 0 \\ 0, & \text{otherwise.} \end{cases}, \tag{5}$$

where in the second case, $\mathbb{E}[S]$ represents the mean service time, and $\rho = \mathbb{E}[S] \times (\mathcal{A}_{i,x,1}^{T,m} + \mathcal{A}_{i,x,2}^{T,m})$ represents the utilization of the queue. The delay can be calculated by plugging in the following Eq. (6a)-(6c):

$$\mathbb{E}[S] = \frac{\alpha_1}{\mu_1} + \frac{\alpha_2}{\mu_2};\ \mathbb{E}\left[S^2\right] = 2\left(\frac{\alpha_1}{(\mu_1)^2} + \frac{\alpha_2}{(\mu_2)^2}\right), \tag{6a}$$

$$\mu_1 = \frac{R_{i,x}^{T,m}}{\mathcal{S}_{i,x,1}^{T,m}};\ \alpha_1 = \frac{\mathcal{A}_{i,x,1}^{T,m}}{\mathcal{A}_{i,x,1}^{T,m} + \mathcal{A}_{i,x,2}^{T,m}}, \tag{6b}$$

$$\mu_2 = \frac{R_{i,x}^{T,m}}{\mathcal{S}_{i,x,2}^{T,m}};\ \alpha_2 = \frac{\mathcal{A}_{i,x,2}^{T,m}}{\mathcal{A}_{i,x,1}^{T,m} + \mathcal{A}_{i,x,2}^{T,m}}. \tag{6c}$$

Eq. (5) must satisfy $D_{i,x}^{T,m} > 0$, so the lower bound $R_{i,x}^{T,m}$ can be derived as

$$LB\_R_{i,x}^{T,m} = \begin{cases} \mathcal{A}_{i,x,1}^{T,m}\mathcal{S}_{i,x,1}^{T,m}, & \text{if } \mathcal{A}_{i,x,1}^{T,m} > 0, \mathcal{A}_{i,x,2}^{T,m} \le 0 \\ \mathcal{A}_{i,x,1}^{T,m}\mathcal{S}_{i,x,1}^{T,m} + \mathcal{A}_{i,x,2}^{T,m}\mathcal{S}_{i,x,2}^{T,m}, & \text{if } \mathcal{A}_{i,x,1}^{T,m} > 0, \mathcal{A}_{i,x,2}^{T,m} > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Hence, the average delay of one data point in training process can be calculated as

$$D_i^{T,m} = \sum_{x \in \mathcal{I}} D_{i,x}^{T,m} + \sum_{n \in \mathcal{I}^{\mathcal{N}}} Z^n. \tag{8}$$

3) Waiting Time in Training Process: We have defined the parameters B_1, B_2, B_3, and B_4 to compute the waiting time for one data point on the training server, first aggregation server, second aggregation server, and *K*-tenant aggregation server, respectively. We denote the waiting times as $D_{i,t}^{W,m}$, $D_{i,f1}^{W,m}$, $D_{i,f2}^{W,m}$, and $D_{i,f3}^{W,m}$, respectively.

To illustrate the calculation of $D_{i,t}^{W,m}$, which represents the average waiting time for one data point at the training server, we note that after training B_1 data points, we perform parameter aggregation. Therefore, the first data point needs to wait for $B_1 - 1$ data points to complete training, and the second data point needs to wait for $B_1 - 2$ data points, and so on. Thus, we can use $\frac{B_1-1}{2}$ to represent the average number to wait for one data point, then divide by the arrival rate to obtain the waiting time. Using this method, we can calculate $D_{i,t}^{W,m}$, $D_{i,f1}^{W,m}$, $D_{i,f2}^{W,m}$, and $D_{i,f3}^{W,m}$ as

$$D_{i,t}^{W,m} = \frac{B_1 - 1}{2} \times \frac{1}{\mathcal{A}_{i,t,2}^{T,m}}, \tag{9a}$$

$$D_{i,f1}^{W,m} = \frac{B_2 - 1}{2} \times \frac{1}{\mathcal{A}_{i,f1,2}^{T,m}}, \tag{9b}$$

$$D_{i,f2}^{W,m} = \frac{B_3 - 1}{2} \times \frac{1}{\mathcal{A}_{i,f2,2}^{T,m}}, \tag{9c}$$

$$D_{i,f3}^{W,m} = \frac{B_4 - 1}{2} \times \frac{1}{\mathcal{A}_{i,f3,2}^{T,m}}, \tag{9d}$$

where $t, f1, f2$, and $f3$ represent the training, first aggregation, second aggregation, and *K*-tenant aggregation servers, respectively.

E. Total Resource Cost Minimization Problem

We focus on minimizing the total resource cost of the Cloud-Edge-Fog service architecture in service scenario *m*, including the computation and communication resource costs during both inference and training processes.

1) Inference and Training Resource cost: Both inference and training processes involve communication and computation resource costs. To calculate the inference resource cost, we define W^{FE}, W^{EF}, W^{EC}, W^{CE}, W^{FF}, and W^{EE} as the communication costs (in money unit/Mbps) of network links. Thus, the communication cost of the *i*-th service architecture in the *m* service scenario during the inference process can be determined as

$$\begin{aligned} N\_C_i^{P,m} = MN \times \left(W^{FE} \times R_{i,FE}^{P,m} + W^{EF} \times R_{i,EF}^{P,m} \right) \\ + M \times \left(W^{EC} \times R^{P,m} + W^{CE} \times R_{i,CE}^{P,m} \right) \\ + \frac{M(M-1)}{2} \times W^{FF} \times R_{i,FF}^{P,m} \\ + \frac{N(N-1)}{2} \times W^{EE} \times R_{i,EE}^{P,m}. \end{aligned} \tag{10}$$

Let W^F, W^E, and W^C be the computation cost (in money unit/MBps) of servers. Thus, the communication cost of *i*-th service architectures in *m* service scenario during inference process is

$$\begin{aligned} S\_C_i^{P,m} = MN \times W^F \times R_{i,F}^{P,m} + M \times W^E \times R_{i,E}^{P,m} \\ + W^C \times R_{i,C}^{P,m}. \end{aligned} \tag{11}$$

Thus, inference resource cost of *i*-th service architectures in *m* service scenario is defined as

$$C_i^{P,m} = N\_C_i^{P,m} + S\_C_i^{P,m}. \tag{12}$$

In addition, $N\_C_i^{T,m}$ and $S\_C_i^{T,m}$ can also be calculated by plugging in $R_{i,x}^{T,m}$, $x \in \mathcal{I}^{\mathcal{N}}$ and $R_{i,x}^{T,m}$, $x \in \mathcal{I}^{\mathcal{S}}$ into Eq. (10) and Eq. (11), respectively. Hence, the training resource cost of the *i*-th service architecture in the *m* service scenario can be derived as

$$C_i^{T,m} = N\_C_i^{T,m} + S\_C_i^{T,m}. \tag{13}$$

2) Total Resource Cost: The total resource cost of *i*-th service architecture in *m* service scenario is defined as

$$\mathbf{C}_i^m = C_i^{P,m} + C_i^{T,m}. \tag{14}$$

Since we aim to minimize the total resource cost of *i*-th service architecture in *m* service scenario while guaranteeing the delay constraints. Therefore, the optimization problem can be mathematically formulated as

$$\begin{aligned} (\mathcal{P}) \quad & \min_{R_{i,x}^{P,m}, R_{i,x}^{T,m}} \mathbf{C}_i^m \\ & C_1 : D_{i,x}^{P,m} < \delta^P \\ & C_2 : D_{i,x}^{T,m} < \delta^T \\ & C_3 : R_{i,x}^{P,m} > LB\_R_{i,x}^{P,m} \\ & C_4 : R_{i,x}^{T,m} > LB\_R_{i,x}^{T,m}. \end{aligned} \tag{15}$$

In Eq. (15), constraint C_1 and C_2 guarantee that $D_{i,x}^{P,m}$ and $D_{i,x}^{T,m}$ respectively satisfy the delay constraints δ^P and δ^T. Constraints C_3 and C_4 are resource constraints which ensure that the resources allocated to each queue must be greater than the lower bound to make $D_{i,x}^{P,m} > 0$ and $D_{i,x}^{T,m} > 0$.

IV. Algorithm Solutions

To address the problems above, we propose the CDRA algorithm to approximate an optimal solution. Additionally, we utilize the Simulated Annealing algorithm for iterative search, as the training and inference processes have unique characteristics, necessitating separate determination of $C_i^{T,m}$ and $C_i^{P,m}$ values.

Algorithm 1: Cost and Delay Resource Allocation (CDRA) Algorithm

Require: $\mathcal{S}_{i,x,l}^{T,m}$, $\mathcal{A}_{i,x,l}^{T,m}$, $LB\_R_{i,x}^{T,m}$, δ^T, W^x, R_unit
Ensure: $C_i^{T,m}$
1: **Initialization**
2: $R_{i,x}^{T,m} \leftarrow R\_LB_{i,x}^{T,m} + R_{unit}$ for all $x \in \mathcal{I}$;
3: $D_i^{T,m} \leftarrow$ Substituting $R_{i,x}^{T,m}$ for all $x \in \mathcal{I}$ into (8);
4: **Find optimal allocated resources by cost value**
5: **while** $D_i^{T,m} > \delta^T$ **do**
6: **for** each queue $x \in \mathcal{I}$ **do**
7: $QD_{before} \leftarrow$ Substituting $R_{i,x}^{T,m}$ into (5);
8: $QD_{after} \leftarrow$ Substituting $R_{i,x}^{T,m} + R_{unit}$ into (5);
9: $CV^x \leftarrow \frac{QD_{before} - QD_{after}}{W^x \times Num^x \times R_{unit}}$;
10: **end for**
11: $choice \leftarrow \arg\max_x CV^x$;
12: $R_{i,choice}^{T,m} \leftarrow R_{i,choice}^{T,m} + R_{unit}$;
13: Update $D_i^{T,m}$ with $R_{i,x}^{T,m}$ for all $x \in \mathcal{I}$;
14: **end while**
15: $C_i^{T,m} \leftarrow$ Substituting $R_{i,x}^{T,m}$ for all $x \in \mathcal{I}$ into (13);

A. Cost and Delay Resource Allocation (CDRA) Algorithm

The CDRA algorithm's core idea is to keep allocating a unit of resource to one of the communication links or computation nodes that can reduce the delay most and cost less until the delay constraint is satisfied.

Algorithm 1 presents the pseudocode of the CDRA algorithm, which we illustrate by deriving $C_i^{T,m}$ as an example. First, we allocate the initial resources to $R_{i,x}^{T,m}$ and calculate the delay (lines 1-2). Then, for each round of resource allocation, we compute the Cost Value CV^x for each queue x (lines 4-8). The queue with the highest CV^x is selected to receive R_{unit}, and we update the delay (lines 9-11). This process continues until $D_{i,x}^{T,m}$ satisfies the delay constraint δ^T. Finally, we compute $C_i^{T,m}$ using Eq. (13).

As to get the value of $C^{P,m}i$, we replace the input parameters of Algorithm 1 ($\mathcal{S}_{i,x,l}^{T,m}$, $\mathcal{A}_{i,x,l}^{T,m}$, $LB\_R_{i,x}^{T,m}$, and δ^T) with ($\mathcal{S}_{i,x,l}^{P,m}$, $\mathcal{A}_{i,x,l}^{P,m}$, $LB\_R_{i,x}^{P,m}$, and δ^P), and replace Eq. (5), Eq. (8), and Eq. (13) with Eq. (1), Eq. (4), and Eq. (12).

B. Simulated Annealing Algorithm

Simulated Annealing (SA) [18] is a global search optimization technique which uses a temperature parameter to manage the randomness of its search, with the search radius converging towards the global optimum as the temperature drops. The algorithm's pseudocode is shown in Algorithm 2. Notably, UB_R_x is the upper bound resource for the x queue ($x \in \mathcal{I}$), while T_i and T_f are the initial and final temperatures, respectively. *MaxIteration* specifies iteration count at each temperature, and γ is the cooling coefficient.

In Algorithm 2, we begin by initializing $R^{T,m}i, x$ with random values between $LB{-}R^{T,m}i, x$ and $UB{-}R_x$, and setting the initial temperature T_i (lines 1–2). During each

Algorithm 2: Simulated Annealing for Resource Allocation

Require: $\mathcal{S}_{i,x,l}^{T,m}$, $\mathcal{A}_{i,x,l}^{T,m}$, $LB\_R_{i,x}^{T,m}$, UB_R_x, δ^T, W^x, R_unit, T_i, T_f, *MaxIteration*, γ
Ensure: $C_i^{T,m}$
1: **Initialization**
2: $R_{i,x}^{T,m} \leftarrow \text{random}(LB\_R_{i,x}^{T,m}, UB\_R_x)$ for all $x \in \mathcal{I}$;
3: $T \leftarrow T_i$;
4: **Iterative searching optimal allocated resources**
5: **while** $T > T_f$ **do**
6: **for** i to *MaxIteration* **do**
7: $AR_{i,x}^{T,m} \leftarrow R_{i,x}^{T,m}$ for all $x \in \mathcal{I}$;
8: $choice \leftarrow$ chooseoneofqueuesinrandom;
9: $AR_{i,choice}^{T,m} \leftarrow \max(\min(R_{i,choice}^{T,m} + \text{random}(-T, T), UB\_R_{choice}), LB\_R_{i,choice}^{T,m})$;
10: $D_i^{T,m} \leftarrow$ substituting $AR_{i,x}^{T,m}$ for all $x \in \mathcal{I}$ into (8);
11: **if** $D_i^{T,m} < \delta^T$ **then**
12: $AC_i^{T,m} \leftarrow$ Substituting $AR_{i,x}^{T,m}$ for all $x \in \mathcal{I}$ into (12);
13: $C_i^{T,m} \leftarrow$ Substituting $R_{i,x}^{T,m}$ for all $x \in \mathcal{I}$ into (12);
14: $\triangle \leftarrow AC_{i,x}^{T,m} - C_{i,x}^{T,m}$;
15: **if** $\triangle < 0$ or $exp(-\triangle / T) > \text{random}(0, 1)$ **then**
16: Update $R_{i,choice}^{T,m}$ with $AR_{i,choice}^{T,m}$;
17: **end if**
18: **end if**
19: **end for**
20: $T \leftarrow T \times \gamma$;
21: **end while**
22: $C_i^{T,m} \leftarrow$ Substituting $R_{i,x}^{T,m}$ for all $x \in \mathcal{I}$ into (12);

temperature iteration (lines 3–6), we iterate *MaxIteration* times, randomly selecting a queue to explore. The resource search space for the queue is determined by the range of $R^{T,m}i, x \pm T$, ensuring values stay within $LB{-}R^{T,m}i, x$ and $UB{-}R_x$ bounds (line 7). After computing the delay (line 8), if it meets the delay constraint, we compare its cost to the current optimum. The allocation is updated based on cost or a probabilistic criterion $exp(-\triangle / T) > random(0, 1)$ (lines 9–14). The temperature is reduced using a cooling efficiency γ (line 18) and the process ends when $T \leq T_f$. We then compute $C_i^{T,m}$ using Eq. (13) (line 20), with $C_i^{P,m}$ derived similarly as in Algorithm 1.

V. Performance Evaluation

In this chapter, we analyze resource allocation performance using the CDRA and SA methods. We also explore the characteristics of different service architectures and scenarios. Our discussion is organized into three main topics: comparing resource allocation algorithms, examining service architectures in NS scenarios, and evaluating the performance across NS, DS, and PS scenarios.

A. Simulation Parameter Setting

In our simulation experiments, we served 10 tenants with a three-tier network architecture comprising one Cloud and

TABLE VI
NETWORK TOPOLOGY IN EXPERIMENTS

Topology	Value	Distance	Value
The number of tenants (K)	15	Fog to Edge	1000m
The number of Edges per tenant (M)	6	Edge to Cloud	100000m
The number of Fogs per Edge (N)	3	Fog to Fog	5m
		Edge to Edge	50m

6 Edges ($M = 6$). Each Edge covered 3 Fogs ($N = 3$). The distance between the network topology and servers is presented in Table VI.

Table VII outlines our experimental setup. We simulated training and inference processes with a data arrival rate of 150 raw data/s from IoT devices. The training dataset and model used were CIFAR-10 [19] and MobilenetV2 [20], respectively. A single task's average input size was 15 Kbit, with the model parameter size at 10 Kbit. The computational demands for training and inference were 30 cycles/bit [21] and 3/bit, respectively. Aggregation in FL and DL required 0.5 cycles/bit. Batch sizes B_1, B_2, B_3, and B_4 were 60, 10, 1, and 10 respectively. The distances between nodes were referenced from the settings in [22]. Delay constraints were 5 and 0.5 seconds for training and inference, respectively.

The following section outlines resource costs and settings for network links and servers. We used the ratio between communication and computational resource costs from [23]. The communication cost per bit rises with increased transmission distance, as per Azure's pricing [24]. Computation costs were drawn from [25], indicating lower costs for higher-tier servers. Hence, the cloud has the lowest computational price, while the Fog's is the highest. Maximum slice resources for fog, edge, and cloud are 1500 MHz, 3000 MHz, and 5000 MHz respectively. EF, FE, and FF have a limit of 100 Mbps, while EC, CE, and EE are set to $100 \times N$ Mbps.

The simulations were conducted using a proprietary simulator developed specifically for this research. The simulator is written in C, tailored to accurately model the complex interactions within AIoT service architectures under various computational and communication constraints.

B. Analyzing Different Resource Allocation Algorithms

We compared the performance of our proposed CDRA algorithm for allocating the resources with SA in the NS scenario as the example. Fig. 4 presents the resource cost allocation and algorithm running time obtained by CDRA and SA. Our findings reveal that the resource cost obtained by CDRA closely approximated the optimal solution achieved by SA's global search. CDRA also demonstrates faster computational performance, requiring less than 1 second compared to SA's average computation time of 270 seconds. This efficiency can be attributed to CDRA's utilization of cost value computation for effective resource allocation, whereas SA requires multiple iterations to find the optimal solution. These findings clearly prove that CDRA is highly effective at optimizing resource allocation in AIoTaS architectures.

However, one notable limitation of the CDRA algorithm arises as the number of nodes within the network architecture

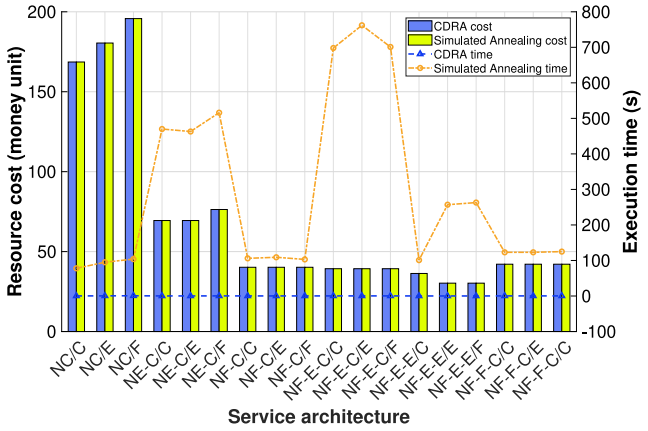


Fig. 4. Compare the training resource cost and execution time calculated by CDRA and Simulated Annealing.

increases. This scalability issue enhances the complexity of the CDRA algorithm, potentially affecting the speed and efficiency of converging to an optimal solution. The increase in complexity stems from the need to manage more resource allocation decision points, as each additional node requires a tailored resource configuration to ensure optimal performance.

Despite this limitation, the iterative nature of the CDRA algorithm provides a notable advantage in highly dynamic environments where system parameters and network conditions may change frequently. This adaptability allows CDRA to adjust resource allocations in real time, effectively responding to network load and configuration variations. Such flexibility is particularly beneficial in variable application scenarios, enhancing the algorithm's utility by addressing potential performance fluctuations and maintaining service quality across different operational conditions.

In summary, while the CDRA algorithm may face challenges in highly complex or rapidly expanding network architectures, its iterative and adaptive capabilities make it well-suited for dynamic environments where flexibility and responsiveness are critical.

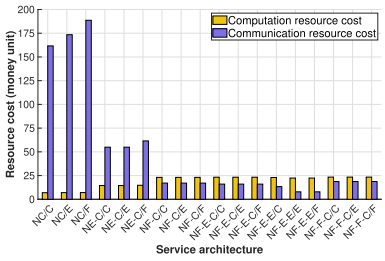
C. Studying Various Service Architectures in NS Scenarios

This section explores the advantages and disadvantages of different service architectures. Since the three service scenarios share similar differences in service architecture, we focus on the NS scenario and divide the analysis into five main themes for discussion.

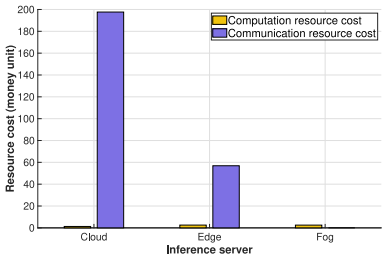
1) Comparing Resource Costs of Different Training Modes and Inference Positions: Figures 5(a) and 5(b) show the communication and computation resource costs for various training modes and inference locations. Figure 5(a) reveals that centralized training incurs communication costs four times higher than other modes due to the need to upload raw data to the cloud. The NE-C architecture, which sends raw data to the edge for training, has slightly lower costs than centralized training. In Figure 5(b), the main difference in inference cost among Cloud, Edge, and Fog is due to communication costs, with the cloud being the most expensive.

TABLE VII
EXPERIMENT PARAMETERS

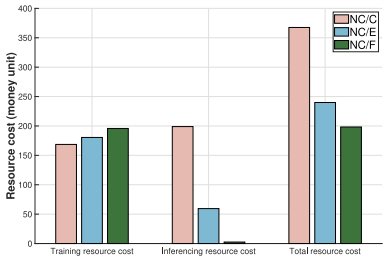
Notation	Meaning	Value	Unit
λ	Arrival rate of raw data collected by each tenant	150	raw data/s
μ_1^L, μ_2^L	Average data size (Exponential)	15, 10	Kbits
μ_1^T, μ_1^P, μ_1^F, μ_2^F, μ_3^F	Average workload (Exponential)	30, 3, 0.5, 0.5, 0.5	cycles/bit
B_1, B_2, B_3, B_4	Model training rate	60, 10, 1, 10	datapoint
δ^T, δ^F	Delay constraint	5, 0.5	second
W^{FE}, W^{EF}, W^{EC}, W^{CE}, W^{FF}, W^{EE}	Communication resource cost	20, 20, 50, 50, 5, 30	money unit/Mbps
W^F, W^E, W^C	Computation resource cost	0.3, 0.2, 0.1	money unit/MHz
UB_R_F, UB_R_E, UB_R_C	Maximum slice resource of servers for a tenant	1500, 3000, 5000	MHz
UB_R_{FE}, UB_R_{EF}, UB_R_{EC}, UB_R_{CE}, UB_R_{EE}, UB_R_{FF}	Maximum slice resource of network links for a tenant	100, 100, $100N$, $100N$, $100N$, 100	Mbps



(a) Training resource cost of service architectures

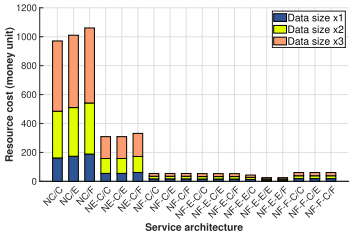


(b) Inference resource cost of different inference servers

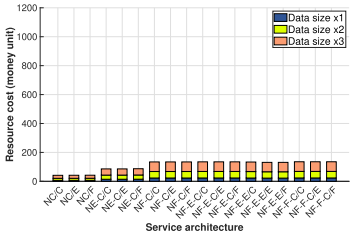


(c) The resource cost of NC/C, NC/E, and NC/F

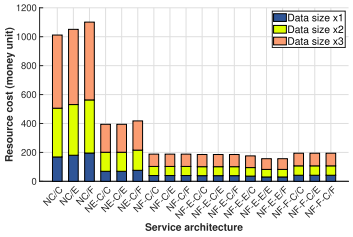
Fig. 5. Compare resource cost among service architectures in the NS service scenario.



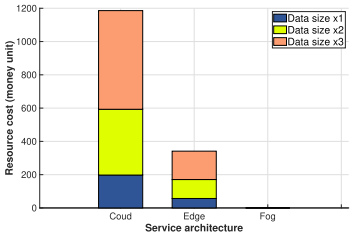
(a) Training communication resource cost



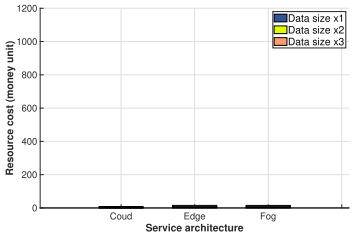
(b) Training computation resource cost



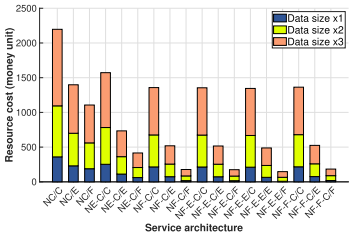
(c) Training total resource cost



(d) Inference communication resource cost



(e) Inference computation resource cost

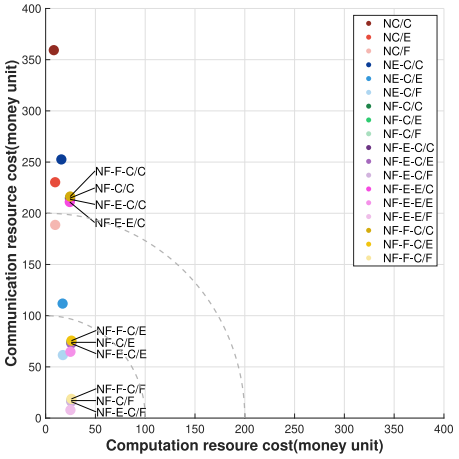


(f) Total resource cost

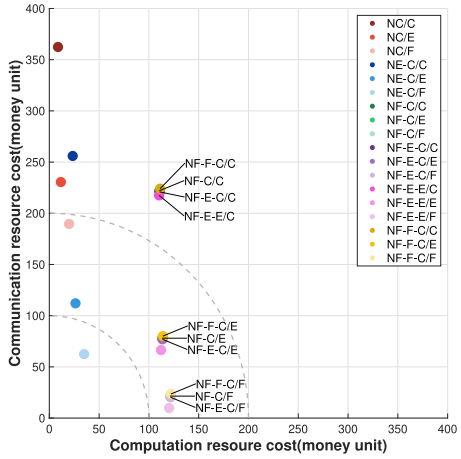
Fig. 6. The effect of different raw data size on service architectures.

Additionally, Figure 5(c) indicates that although NC/C has lower training costs, the high communication cost of cloud-based inference negates its benefits. Thus, fog-based inference is more advantageous, potentially saving up to 44% in resource costs.

2) Comparison of Raw Data Sizes: This subsection examines how varying raw data sizes impact service architectures. Figures 6(a), 6(b), and 6(c) display the training communication, computation, and total resource costs as data sizes increase from one to three times the original size. The results show a significant increase in communication costs for centralized training (NS) compared to distributed learning (NE-C), with costs for NS scenarios rising up to five times. This suggests that transmitting larger raw data to remote servers becomes increasingly inefficient. Figures 6(d) and 6(e) demonstrate the inference communication and computation costs across cloud, edge, and fog, indicating rising communication costs with larger data sizes. Figure 6(f) presents total resource costs for each architecture, highlighting that fog scenarios, especially NF-E-E/F, maintain relative cost stability despite increasing data sizes, making them the most resource-efficient options.



(a) Server cost (Fog:Edge:Cloud = 3:2:1)



(b) Server cost (Fog:Edge:Cloud = 15:3:1)

Fig. 7. The effect of different server cost on service architectures.

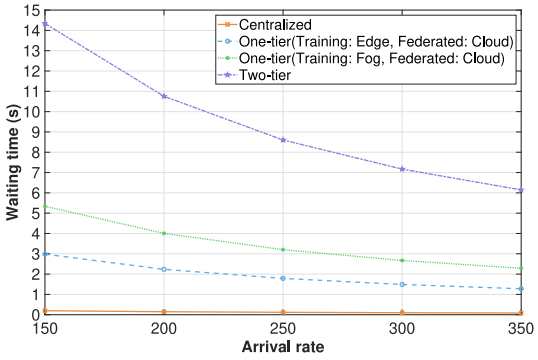
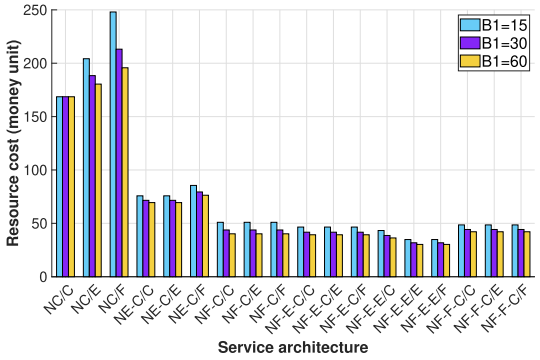
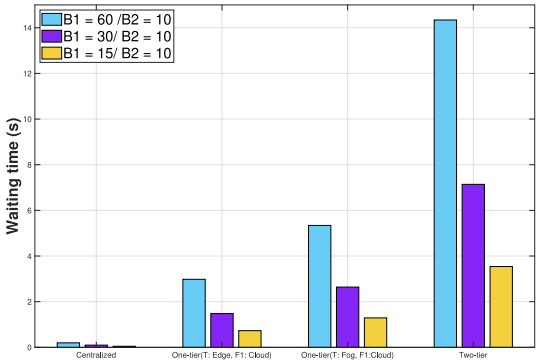


Fig. 8. Waiting time of different training hierarchical architecture.



(a) Training resource cost of different B_1



(b) Waiting time of different B_1

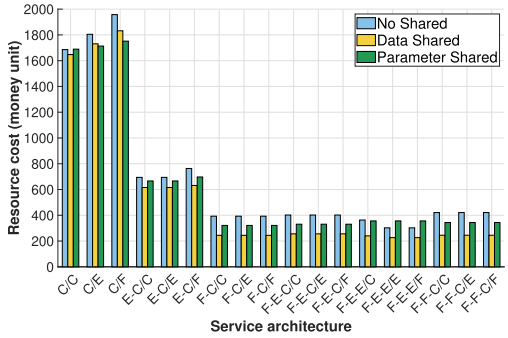
Fig. 9. The effect of setting different B_1 on training resource cost and waiting time.

3) Exploring the Impact of Server Cost on Service Architectures: We analyze training resource costs among various service architectures with different server cost settings. Figure 7(a) shows that architectures like NC/C and NC/E, which perform inference in the cloud, incur higher total resource costs. Conversely, architectures like NE-C/F, which involve training in the fog and inference in the edge and fog, exhibit lower costs. Adjusting the server cost ratio of Fog:Edge:Cloud to 15:3:1, as shown in Figure 7(b), highlights even greater cost disparities. Under these settings, training in the fog may not be cost-effective, making the NE-C/F architecture the most economical option with the lowest total resource costs.

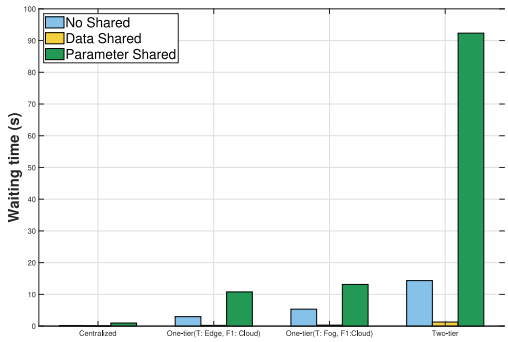
4) Comparing the Waiting Time of Different Service Architectures: In this subsection, we analyze the waiting time of various service architectures. We assess the waiting time for different training modes, encompassing Centralized, One-tier (Training at Fog or Training at Edge), and Two-tier, while considering different values of λ.

Figure 8 shows that higher λ values lead to reduced waiting times, indicating that data points spend less time waiting as the data collection rate increases. The waiting times rank from longest to shortest as follows: two-tier, one-tier, and centralized. Training modes involving more aggregation iterations incur longer waiting times. Notably, one-tier training at the fog experiences an 83% longer waiting time compared to training at the edge, attributed to the higher data collection rate at the edge.

5) Impact of B_1 on Waiting Time in Different Service Architectures: This subsection examines how different batch sizes (B_1) affect service architectures. Fig. 9 shows the training resource cost and waiting time for B_1 values of 15, 30, and 60. As illustrated in Fig. 9(a), larger B_1 reduces training resource costs. This is because bigger batches lead to fewer parameter updates, decreasing resource needs at the



(a) Training resource cost of different service scenarios



(b) Waiting time of different service scenarios

Fig. 10. Compare training resource cost and waiting time of three service scenarios.

aggregation server. However, Fig. 9(b) indicates that a larger B_1 increases waiting times during training. Thus, choosing B_1 requires balancing resource costs and waiting times based on the application's priority.

D. Comparing the Performance of NS, DS, and PS Service Scenarios

In this subsection, we evaluate the training resource costs and waiting times for NS, DS, and PS modes involving K tenants. For NS, we multiply the cost for one tenant by K to facilitate an accurate comparison. In PS, the training costs for individual tenants are similarly scaled by K. For DS, where tenants jointly train their models, the aggregated cost for K tenants is considered.

Setting K and B_3 both to 10, Figure 10(a) shows that NS typically incurs the highest training resource costs, more than triple those of other scenarios. In contrast, DS is the most resource-efficient scenario. NC/E and NC/F scenarios, which involve transmitting trained parameters to the inference server, have training resource costs that are about 5.8% and 9.25% higher, respectively, compared to PC+C/E and PC+C/F scenarios. This cost difference is due to PC+C/E and PC+C/F scenarios benefiting from parameter aggregation from K tenants, which reduces the volume of data transmitted.

Figure 10(b) shows waiting times for three service scenarios: Centralized, One-tier (training at Fog or Edge), and Two-tier. The PS scenario shows significantly longer waiting times, often two to six times that of other scenarios, due to the need for aggregating data from an additional K tenants. In contrast, the DS scenario records the shortest waiting times, as collaborative training among K tenants speeds up data aggregation.

TABLE VIII
COMPARISON OF MODEL ACCURACY ACROSS THREE SERVICE SCENARIOS

Scenario		Test accuracy
No Shared	5000 images/tenant	77.04%
	10000 images/tenant	85.32%
Data Shared		93.88%
Parameter Shared		88.55%

Furthermore, we compared model accuracy across three service scenarios using the MobileNet V2 model on the cifar-10 dataset. The dataset was splitting into 50,000 training and 10,000 testing images. Each tenant received a random 5,000-image allocation. Table VIII displays the test accuracy results.

The DS scenario had the highest accuracy due to collaborative training across all tenants. In the PS scenario, local models from all tenants were merged into a global model using the FedAvg method, yielding slightly reduced accuracy. The NS scenario's accuracy was 77.04% because of limited training data from each tenant's 5,000 images. However, doubling the images per tenant raised accuracy to 85.32%. This demonstrates that more tenant data can lead to increased accuracy.

As mentioned above, the advantages and disadvantages of the three service scenarios are summarized. Although NS provides the best privacy protection, most service architectures require the highest training resource cost. In contrast, DS ignores privacy, requires the least resource cost, and has the shortest average waiting time during training. It also can achieve the highest accuracy. PS provides both privacy protection and collaborative training of K-tenant data through K-tenant aggregation but has the longest average waiting time.

E. Lessons Learned for Service Providers

For service providers, our research findings provide the following valuable insights.

- *Fog or Edge for Training and Inference.* It is recommended to conduct training and inference at the Fog or Edge levels. Service providers should evaluate the volume of raw data and the cost discrepancies across cloud, edge, and fog computing to select the most suitable service architecture. If significant cost differences exist, an architecture that integrates edge training with fog inference may offer a more cost-effective solution.
- *Parameter Aggregation Can Be Time-consuming.* Service providers should note that while fog computing often reduces resource costs, it increases communication overhead due to frequent data collection and transmission. It is essential to balance service delivery to ensure efficient execution times and optimal resource utilization.
- *Trade-off Between Privacy and Training Cost.* Service providers should evaluate the differing data privacy requirements and operational costs across service scenarios. Most AIoTaS applications are likely to be implemented as large-scale public services, except for

TABLE IX
THE ARRIVAL RATE AND SERVICE DEMAND PARAMETERS WITH DIFFERENT SERVICE ARCHITECTURES IN NS DURING TRAINING PROCESS

Service Architectures	$\mathcal{A}/\mathcal{S}$	**ID**	**Communication (data size)**						**Computation (workload)**		
			FE	EC	CE	EF	FF	EE	F	E	C
NC/C	$\mathcal{A}$	i=1	λ/MN	λ/M	0	0	0	0	0	0	λ
	$\mathcal{S}$		μ_1^L	μ_1^L	0	0	0	0	0	0	μ_1^T
NC/E	$\mathcal{A}$	i=2	λ/MN	λ/M	λ/B_1	0	0	0	0	0	λ
	$\mathcal{S}$		μ_1^L	μ_1^L	μ_2^L	0	0	0	0	0	μ_1^T
NC/F	$\mathcal{A}$	i=3	λ/MN	λ/M	λ/B_1	λ/B_1	0	0	0	0	λ
	$\mathcal{S}$		μ_1^L	μ_1^L	μ_2^L	μ_2^L	0	0	0	0	μ_1^T
NE-C/C	$\mathcal{A}$	i=4	λ/MN	λ/MB_1	λ/B_1B_2	0	0	0	0	λ/M	λ/B_1
	$\mathcal{S}$		μ_1^L	μ_2^L	μ_2^L	0	0	0	0	μ_1^T	μ_2^F
NE-C/E	$\mathcal{A}$	i=5	λ/MN	λ/MB_1	λ/B_1B_2	0	0	0	0	λ/M	λ/B_1
	$\mathcal{S}$		μ_1^L	μ_2^L	μ_2^L	0	0	0	0	μ_1^T	μ_2^F
NE-C/F	$\mathcal{A}$	i=6	λ/MN	λ/MB_1	λ/B_1B_2	λ/B_1B_2	0	0	0	λ/M	λ/B_1
	$\mathcal{S}$		μ_1^L	μ_2^L	μ_2^L	μ_2^L	0	0	0	μ_1^T	μ_2^F
NF-C/C	$\mathcal{A}$	i=7	λ/MNB_1	λ/MB_1	λ/B_1B_2	λ/B_1B_2	0	0	λ/MN	0	λ/MN
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	0	μ_1^T	0	μ_2^F
NF-C/E	$\mathcal{A}$	i=8	λ/MNB_1	λ/MB_1	λ/B_1B_2	λ/B_1B_2	0	0	λ/MN	0	λ/MN
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	0	μ_1^T	0	μ_2^F
NF-C/F	$\mathcal{A}$	i=9	λ/MNB_1	λ/MB_1	λ/B_1B_2	λ/B_1B_2	0	0	λ/MN	0	λ/MN
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	0	μ_1^T	0	μ_2^F
NF-E-C/C	$\mathcal{A}$	i=10	λ/MNB_1	λ/MB_1B_2	$\lambda/B_1B_2B_3$	$\lambda/B_1B_2B_3$	0	0	λ/MN	λ/MB_1	λ/B_1B_2
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	0	μ_1^T	μ_2^F	μ_2^F
NF-E-C/E	$\mathcal{A}$	i=11	λ/MNB_1	λ/MB_1B_2	$\lambda/B_1B_2B_3$	$\lambda/B_1B_2B_3$	0	0	λ/MN	λ/MB_1	λ/B_1B_2
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	0	μ_1^T	μ_2^F	μ_2^F
NF-E-C/F	$\mathcal{A}$	i=12	λ/MNB_1	λ/MB_1B_2	$\lambda/B_1B_2B_3$	$\lambda/B_1B_2B_3$	0	0	λ/MN	λ/MB_1	λ/B_1B_2
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	0	μ_1^T	μ_2^F	μ_2^F
NF-E-E/C	$\mathcal{A}$	i=13	λ/MNB_1	$\lambda/B_1B_2B_3$	0	$\lambda/B_1B_2B_3$	0	λ/MB_1B_2	λ/MN	λ/MB_1, λ/B_1B_2	0
	$\mathcal{S}$		μ_2^L	μ_2^L	0	μ_2^L	0	μ_2^L	μ_1^T	μ_2^F, μ_1^F	0
NF-E-E/E	$\mathcal{A}$	i=14	λ/MNB_1	0	0	$\lambda/B_1B_2B_3$	0	λ/MB_1B_2	λ/MN	λ/MB_1, λ/B_1B_2	0
	$\mathcal{S}$		μ_2^L	0	0	μ_2^L	0	μ_2^L	μ_1^T	μ_2^F, μ_1^F	0
NF-E-E/F	$\mathcal{A}$	i=15	λ/MNB_1	0	0	$\lambda/B_1B_2B_3$	0	λ/MB_1B_2	λ/MN	λ/MB_1, λ/B_1B_2	0
	$\mathcal{S}$		μ_2^L	0	0	μ_2^L	0	μ_2^L	μ_1^T	μ_2^F, μ_1^F	0
NF-F-C/C	$\mathcal{A}$	i=16	λ/MB_1B_2	λ/MB_1B_2	$\lambda/B_1B_2B_3$	$\lambda/B_1B_2B_3$	λ/MNB_1	0	λ/MN, λ/MB_1	0	λ/B_1B_2
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	μ_1^T, μ_1^F	0	μ_2^F
NF-F-C/E	$\mathcal{A}$	i=17	λ/MB_1B_2	λ/MB_1B_2	$\lambda/B_1B_2B_3$	$\lambda/B_1B_2B_3$	λ/MNB_1	0	λ/MN, λ/MB_1	0	λ/B_1B_2
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	μ_1^T, μ_1^F	0	μ_2^F
NF-F-C/F	$\mathcal{A}$	i=18	λ/MB_1B_2	λ/MB_1B_2	$\lambda/B_1B_2B_3$	$\lambda/B_1B_2B_3$	λ/MNB_1	0	λ/MN, λ/MB_1	0	λ/B_1B_2
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	μ_1^T, μ_1^F	0	μ_2^F

data-sensitive, small-scale services using the NS scenario, which may restrict learning capabilities and increase training costs. Operationally, the DS scenario is more cost-effective but offers limited data privacy. Conversely, the PS scenario enhances privacy protection but may result in longer waiting times.

VI. CONCLUSION

This paper categorizes AIoTaS into three service scenarios: NS, DS, and PS, and explores multiple service architectures. Our analysis focuses on the trade-offs between these services regarding cost, data privacy, and accuracy. The objective is to provide actionable recommendations for service providers on optimal AIoTaS configurations. We have formulated a resource optimization problem to minimize resource costs by determining optimal allocations. Leveraging the convexity of this problem, we developed the CDRA method to achieve near-optimal solutions.

A series of experiments were conducted to guide service providers on deploying service architectures effectively. Results indicate that training and inference in fog or edge environments can substantially reduce costs compared to the cloud, with inference in the fog potentially cutting resource costs by up to 44%. Our evaluations revealed that the NS scenario is the most resource-demanding, with training costs more than three times higher than other models, yet it provides robust privacy. Conversely, the DS scenario is the most resource-efficient, significantly lowering costs and achieving a high model accuracy of 93.88%, but at the cost of privacy. The PS scenario offers a balance, providing better privacy but with longer waiting times than the DS scenario.

Several issues need further study. In this paper, we only consider a homogeneous architecture. In the real world, a heterogeneous architecture is expected and needs further investigation. Furthermore, the effective operation of different service scenarios under different network architectures also deserves further study.

TABLE X
THE ARRIVAL RATE AND SERVICE DEMAND PARAMETERS WITH DIFFERENT SERVICE ARCHITECTURES IN DS DURING TRAINING PROCESS

Service Architectures	$\mathcal{A}$/$\mathcal{S}$	**ID**	**Communication (data size)**						**Computation (workload)**		
			FE	EC	CE	EF	FF	EE	F	E	C
DC/C	$\mathcal{A}$	i=1	$\lambda K/MN$	$\lambda K/M$	0	0	0	0	0	0	λK
	$\mathcal{S}$		μ_1^L	μ_1^L	0	0	0	0	0	0	μ_1^T
DC/E	$\mathcal{A}$	i=2	$\lambda K/MN$	$\lambda K/M$	$\lambda K/B_1$	0	0	0	0	0	λK
	$\mathcal{S}$		μ_1^L	μ_1^L	μ_2^L	0	0	0	0	0	μ_1^T
DC/F	$\mathcal{A}$	i=3	$\lambda K/MN$	$\lambda K/M$	$\lambda K/B_1$	$\lambda K/B_1$	0	0	0	0	λK
	$\mathcal{S}$		μ_1^L	μ_1^L	μ_2^L	μ_2^L	0	0	0	0	μ_1^T
DE-C/C	$\mathcal{A}$	i=4	$\lambda K/MN$	$\lambda K/MB_1$	$\lambda K/B_1B_2$	0	0	0	0	$\lambda K/M$	$\lambda K/B_1$
	$\mathcal{S}$		μ_1^L	μ_2^L	μ_2^L	0	0	0	0	μ_1^T	μ_2^F
DE-C/E	$\mathcal{A}$	i=5	$\lambda K/MN$	$\lambda K/MB_1$	$\lambda K/B_1B_2$	0	0	0	0	$\lambda K/M$	$\lambda K/B_1$
	$\mathcal{S}$		μ_1^L	μ_2^L	μ_2^L	0	0	0	0	μ_1^T	μ_2^F
DE-C/F	$\mathcal{A}$	i=6	$\lambda K/MN$	$\lambda K/MB_1$	$\lambda K/B_1B_2$	$\lambda K/B_1B_2$	0	0	0	$\lambda K/M$	$\lambda K/B_1$
	$\mathcal{S}$		μ_1^L	μ_2^L	μ_2^L	μ_2^L	0	0	0	μ_1^T	μ_2^F
DF-C/C	$\mathcal{A}$	i=7	$\lambda K/MNB_1$	$\lambda K/MB_1$	$\lambda K/B_1B_2$	$\lambda K/B_1B_2$	0	0	$\lambda K/MN$	0	$\lambda K/MN$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	0	μ_1^T	0	μ_2^F
DF-C/E	$\mathcal{A}$	i=8	$\lambda K/MNB_1$	$\lambda K/MB_1$	$\lambda K/B_1B_2$	$\lambda K/B_1B_2$	0	0	$\lambda K/MN$	0	$\lambda K/MN$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	0	μ_1^T	0	μ_2^F
DF-C/F	$\mathcal{A}$	i=9	$\lambda K/MNB_1$	$\lambda K/MB_1$	$\lambda K/B_1B_2$	$\lambda K/B_1B_2$	0	0	$\lambda K/MN$	0	$\lambda K/MN$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	0	μ_1^T	0	μ_2^F
DF-E-C/C	$\mathcal{A}$	i=10	$\lambda K/MNB_1$	$\lambda K/MB_1B_2$	$\lambda K/B_1B_2B_3$	$\lambda K/B_1B_2B_3$	0	0	$\lambda K/MN$	$\lambda K/MB_1$	$\lambda K/B_1B_2$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	0	μ_1^T	μ_2^F	μ_2^F
DF-E-C/E	$\mathcal{A}$	i=11	$\lambda K/MNB_1$	$\lambda K/MB_1B_2$	$\lambda K/B_1B_2B_3$	$\lambda K/B_1B_2B_3$	0	0	$\lambda K/MN$	$\lambda K/MB_1$	$\lambda K/B_1B_2$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	0	μ_1^T	μ_2^F	μ_2^F
DF-E-C/F	$\mathcal{A}$	i=12	$\lambda K/MNB_1$	$\lambda K/MB_1B_2$	$\lambda K/B_1B_2B_3$	$\lambda K/B_1B_2B_3$	0	0	$\lambda K/MN$	$\lambda K/MB_1$	$\lambda K/B_1B_2$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	0	μ_1^T	μ_2^F	μ_2^F
DF-E-E/C	$\mathcal{A}$	i=13	$\lambda K/MNB_1$	$\lambda K/B_1B_2B_3$	0	$\lambda K/B_1B_2B_3$	0	$\lambda K/MB_1B_2$	$\lambda K/MN$	$\lambda K/MB_1$, $\lambda K/B_1B_2$	0
	$\mathcal{S}$		μ_2^L	μ_2^L	0	μ_2^L	0	μ_2^L	μ_1^T	μ_2^F, μ_1^F	0
DF-E-E/E	$\mathcal{A}$	i=14	$\lambda K/MNB_1$	0	0	$\lambda K/B_1B_2B_3$	0	$\lambda K/MB_1B_2$	$\lambda K/MN$	$\lambda K/MB_1$, $\lambda K/B_1B_2$	0
	$\mathcal{S}$		μ_2^L	0	0	μ_2^L	0	μ_2^L	μ_1^T	μ_2^F, μ_1^F	0
DF-E-E/F	$\mathcal{A}$	i=15	$\lambda K/MNB_1$	0	0	$\lambda K/B_1B_2B_3$	0	$\lambda K/MB_1B_2$	$\lambda K/MN$	$\lambda K/MB_1$, $\lambda K/B_1B_2$	0
	$\mathcal{S}$		μ_2^L	0	0	μ_2^L	0	μ_2^L	μ_1^T	μ_2^F, μ_1^F	0
DF-F-C/C	$\mathcal{A}$	i=16	$\lambda K/MB_1B_2$	$\lambda K/MB_1B_2$	$\lambda K/B_1B_2B_3$	$\lambda K/B_1B_2B_3$	$\lambda K/MNB_1$	0	$\lambda K/MN$, $\lambda K/MB_1$	0	$\lambda K/B_1B_2$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	μ_1^T, μ_1^F	0	μ_2^F
DF-F-C/E	$\mathcal{A}$	i=17	$\lambda K/MB_1B_2$	$\lambda K/MB_1B_2$	$\lambda K/B_1B_2B_3$	$\lambda K/B_1B_2B_3$	$\lambda K/MNB_1$	0	$\lambda K/MN$, $\lambda K/MB_1$	0	$\lambda K/B_1B_2$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	μ_1^T, μ_1^F	0	μ_2^F
DF-F-C/F	$\mathcal{A}$	i=18	$\lambda K/MB_1B_2$	$\lambda K/MB_1B_2$	$\lambda K/B_1B_2B_3$	$\lambda K/B_1B_2B_3$	$\lambda K/MNB_1$	0	$\lambda K/MN$, $\lambda K/MB_1$	0	$\lambda K/B_1B_2$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	μ_1^T, μ_1^F	0	μ_2^F

APPENDIX

This section provides a detailed description of the service demand and arrival rate parameters for all NS, DS, and PS service scenarios architectures during the training and inference processes. These parameters are presented in Table IX, X, and XI, respectively, and the differences between the DS and PS scenarios and the NS scenario are explained.

We first derive the parameters for the DS service scenario. The main difference between the DS and NS scenarios is that in the DS scenario, a wholly shared mode is adopted during training, where all tenants' collected data is trained together. Taking the DF-E-C/F and NF-E-C/F architectures as examples, the difference in the arrival rate parameter between them is multiplied by a factor of K. This is because in the DS scenario, there is no need to separate the resources used by each tenant through slicing. Therefore, the total resource cost calculated for the DS scenario is the resource cost required by *K* tenants, while in the NS scenario, it is the resource cost needed for a

TABLE XI
THE ARRIVAL RATE AND SERVICE DEMAND PARAMETERS WITH DIFFERENT SERVICE ARCHITECTURES IN PS DURING TRAINING PROCESS

Service Architectures	$\mathcal{A}/\mathcal{S}$	ID	Communication (data size)						Computation (workload)		
			FE	EC	CE	EF	FF	EE	F	E	C
PC+C/C	$\mathcal{A}$	i=1	λ/MN	λ/M	0	0	0	0	0	0	λ, $\lambda K/B_1$
	$\mathcal{S}$		μ_1^L	μ_1^L	0	0	0	0	0	0	μ_1^T, μ_3^F
PC+C/E	$\mathcal{A}$	i=2	λ/MN	λ/M	$\lambda K/B_1/B_4$	0	0	0	0	0	λ, $\lambda K/B_1$
	$\mathcal{S}$		μ_1^L	μ_1^L	μ_2^L	0	0	0	0	0	μ_1^T, μ_3^F
PC+C/F	$\mathcal{A}$	i=3	λ/MN	λ/M	$\lambda K/B_1B_4$	$\lambda K/B_1B_4$	0	0	0	0	λ, $\lambda K/B_1$
	$\mathcal{S}$		μ_1^L	μ_1^L	μ_2^L	μ_2^L	0	0	0	0	μ_1^T, μ_3^F
PE-C+C/C	$\mathcal{A}$	i=4	λ/MN	λ/MB_1	$\lambda K/B_1B_2B_4$	0	0	0	0	λ/M	λ/B_1, $\lambda K/B_1B_2$
	$\mathcal{S}$		μ_1^L	μ_2^L	μ_2^L	0	0	0	0	μ_1^T	μ_2^F, μ_3^F
PE-C+C/E	$\mathcal{A}$	i=5	λ/MN	λ/MB_1	$\lambda K/B_1B_2B_4$	0	0	0	0	λ/M	λ/B_1, $\lambda K/B_1B_2$
	$\mathcal{S}$		μ_1^L	μ_2^L	μ_2^L	0	0	0	0	μ_1^T	μ_2^F, μ_3^F
PE-C+C/F	$\mathcal{A}$	i=6	λ/MN	λ/MB_1	$\lambda K/B_1B_2B_4$	$\lambda/B_1B_2B_4$	0	0	0	λ/M	λ/B_1, $\lambda K/B_1B_2$
	$\mathcal{S}$		μ_1^L	μ_2^L	μ_2^L	μ_2^L	0	0	0	μ_1^T	μ_2^F, μ_3^F
PF-C+C/C	$\mathcal{A}$	i=7	λ/MNB_1	λ/MB_1	$\lambda K/B_1B_2B_4$	$\lambda K/B_1B_2B_4$	0	0	λ/MN	0	λ/B_1, $\lambda K/B_1B_2$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	0	μ_1^T	0	μ_2^F, μ_3^F
PF-C+C/E	$\mathcal{A}$	i=8	λ/MNB_1	λ/MB_1	$\lambda/B_1B_2B_4$	$\lambda/B_1B_2B_4$	0	0	λ/MN	0	λ/B_1, $\lambda K/B_1B_2$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	0	μ_1^T	0	μ_2^F, μ_3^F
PF-C+C/F	$\mathcal{A}$	i=9	λ/MNB_1	λ/MB_1	$\lambda/B_1B_2B_4$	$\lambda/B_1B_2B_4$	0	0	λ/MN	0	λ/B_1, $\lambda K/B_1B_2$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	0	μ_1^T	0	μ_2^F, μ_3^F
PF-E-C+C/C	$\mathcal{A}$	i=10	λ/MNB_1	λ/MB_1B_2	$\lambda K/B_1B_2B_3B_4$	$\lambda K/B_1B_2B_3B_4$	0	0	λ/MN	λ/MB_1	λ/B_1B_2, $\lambda K/B_1B_2B_3$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	0	μ_1^T	μ_2^F	μ_2^F, μ_3^F
PF-E-C+C/E	$\mathcal{A}$	i=11	λ/MNB_1	λ/MB_1B_2	$\lambda K/B_1B_2B_3B_4$	$\lambda K/B_1B_2B_3B_4$	0	0	λ/MN	λ/MB_1	λ/B_1B_2, $\lambda K/B_1B_2B_3$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	0	μ_1^T	μ_2^F	μ_2^F, μ_3^F
PF-E-C+C/F	$\mathcal{A}$	i=12	λ/MNB_1	λ/MB_1B_2	$\lambda K/B_1B_2B_3B_4$	$\lambda K/B_1B_2B_3B_4$	0	0	λ/MN	λ/MB_1	λ/B_1B_2, $\lambda K/B_1B_2B_3$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	0	μ_1^T	μ_2^F	μ_2^F, μ_3^F
PF-E-E+C/C	$\mathcal{A}$	i=13	λ/MNB_1	$\lambda/B_1B_2B_3$	$\lambda K/B_1B_2B_3B_4$	$\lambda K/B_1B_2B_3B_4$	0	λ/MB_1B_2	λ/MN	λ/MB_1, λ/B_1B_2	$\lambda K/B_1B_2B_3$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	μ_2^L	μ_1^T	μ_2^F, μ_1^F	μ_4^L
PF-E-E+C/E	$\mathcal{A}$	i=14	λ/MNB_1	$\lambda/B_1B_2B_3$	$\lambda K/B_1B_2B_3B_4$	$\lambda K/B_1B_2B_3B_4$	0	λ/MB_1B_2	λ/MN	λ/MB_1, λ/B_1B_2	$\lambda K/B_1B_2B_3$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	μ_2^L	μ_1^T	μ_2^F, μ_1^F	μ_3^F
PF-E-E+C/F	$\mathcal{A}$	i=15	λ/MNB_1	$\lambda/B_1B_2B_3$	$\lambda K/B_1B_2B_3B_4$	$\lambda K/B_1B_2B_3B_4$	0	λ/MB_1B_2	λ/MN	λ/MB_1, λ/B_1B_2	$\lambda K/B_1B_2B_3$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	μ_2^L	μ_1^T	μ_2^F, μ_1^F	μ_4^L
PF-F-C+C/C	$\mathcal{A}$	i=16	λ/MB_1B_2	λ/MB_1B_2	$\lambda K/B_1B_2B_3B_4$	$\lambda K/B_1B_2B_3B_4$	λ/MNB_1	0	λ/MN, λ/MB_1	0	λ/B_1B_2 $\lambda K/B_1B_2B_3$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	μ_1^T, μ_1^F	0	μ_2^F, μ_3^F
PF-F-C+C/E	$\mathcal{A}$	i=17	λ/MB_1B_2	λ/MB_1B_2	$\lambda K/B_1B_2B_3B_4$	$\lambda K/B_1B_2B_3B_4$	λ/MNB_1	0	λ/MN, λ/MB_1	0	λ/B_1B_2 $\lambda K/B_1B_2B_3$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	μ_1^T, μ_1^F	0	μ_2^F, μ_3^F
PF-F-C+C/F	$\mathcal{A}$	i=18	λ/MB_1B_2	λ/MB_1B_2	$\lambda K/B_1B_2B_3B_4$	$\lambda K/B_1B_2B_3B_4$	λ/MNB_1	0	λ/MN, λ/MB_1	0	λ/B_1B_2 $\lambda K/B_1B_2B_3$
	$\mathcal{S}$		μ_2^L	μ_2^L	μ_2^L	μ_2^L	μ_2^L	0	μ_1^T, μ_1^F	0	μ_2^F, μ_3^F

single tenant. As mentioned above, the arrival rate parameters for all service architectures in the DS scenario can be derived in this manner, while the service demand parameters are the same as those in the NS scenario.

Next, we derive the parameters for the PS scenario. The main difference between the PS and NS scenarios is that in the PS scenario, an additional FL is performed in the Cloud during training, where *K* tenant model parameters are shared. Taking the PF-E-C+C/F and NF-E-C/F architectures as examples, the difference is that PF-E-C+C/F performs B_3 rounds of aggregation in the Cloud and aggregates the parameters of *K* tenants into a global model. Therefore, the arrival rate for the *K*-tenant aggregation task is $\lambda K/B_1B_2B_3$. After B_4 rounds of *K*-tenant aggregation, the model parameters are updated

in the Fog, so the arrival rate for the CE and EF queues is $\lambda K / B_1 B_2 B_3 B_4$. An additional K-tenant aggregation service demand parameter, μ_3^F, is also required in the Cloud. As stated above, the arrival rate and service demand parameters for all service architectures in the PS scenario can be derived in this manner.

REFERENCES

[1] J. Du, C. Jiang, J. Wang, Y. Ren, and M. Debbah, "Machine learning for 6G wireless networks: Carrying forward enhanced bandwidth, massive access, and ultrareliable/low-latency service," *IEEE Veh. Technol. Mag.*, vol. 15, no. 4, pp. 122–134, Dec. 2020.

[2] K. B. Letaief, Y. Shi, J. Lu, and J. Lu, "Edge artificial intelligence for 6G: Vision, enabling technologies, and applications," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 5–36, Jan. 2022.

[3] F. Firouzi, B. Farahani, and A. Marinšek, "The convergence and interplay of edge, fog, and cloud in the AI-driven Internet of Things (IoT)," *Inf. Syst.*, vol. 107, Jul. 2022, Art. no. 101840.

[4] S. Kekki et al., "MEC in 5G networks," ETSI, Sophia Antipolis, France, White Paper, 2018.

[5] P.-H. Kuo et al., "An integrated edge and fog system for future communication networks," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, 2018, pp. 338–343.

[6] B. Kar, W. Yahya, Y.-D. Lin, and A. Ali, "Offloading using traditional optimization and machine learning in federated cloud-edge-fog systems: A survey," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 2, pp. 1199–1226, 2nd Quart., 2023.

[7] Y. Liang, T. Cui, Y. Cao, and H. Tang, "An effective resource scheduling model for edge cloud oriented AIoT," *Concurr. Comput. Pract. Exp.*, vol. 34, no. 5, 2022, Art. no. e6720.

[8] S. Zhu, K. Ota, and M. Dong, "Energy-efficient artificial intelligence of things with intelligent edge," *IEEE Internet Things J.*, vol. 9, no. 10, pp. 7525–7532, May 2022.

[9] S. Pal, N. Z. Jhanjhi, A. S. Abdulbaqi, D. Akila, A. A. Almazroi, and F. S. Alsubaei, "A hybrid edge-cloud system for networking service components optimization using the Internet of Things," *Electronics*, vol. 12, no. 3, p. 649, Jan. 2023. [Online]. Available: https://doi.org/10.3390/electronics12030649

[10] J. Chen, P. Yang, S. Ren, Z. Zhao, X. Cao, and D. Wu, "Enhancing AIoT device association with task offloading in aerial MEC networks," *IEEE Internet Things J.*, vol. 11, no. 1, pp. 174–187, Jan. 2024.

[11] F. Xu, F. Yang, S. Bao, and C. Zhao, "DQN inspired joint computing and caching resource allocation approach for software defined information-centric Internet of Things network," *IEEE Access*, vol. 7, pp. 61987–61996, 2019.

[12] Y. Hao, Y. Miao, L. Hu, M. S. Hossain, G. Muhammad, and S. U. Amin, "Smart-edge-CoCaCo: AI-enabled smart edge with joint computation, caching, and communication in heterogeneous IoT," *IEEE Netw.*, vol. 33, no. 2, pp. 58–64, Mar./Apr. 2019.

[13] Z. Xiang et al., "Energy-effective artificial Internet-of-Things application deployment in edge-cloud systems," *Peer-to-Peer Netw. Appl.*, vol. 15, no. 2, pp. 1029–1044, Dec. 2021. [Online]. Available: https://doi.org/10.1007/s12083-021-01273-5

[14] X. Ji, J. Tian, H. Zhang, D. Wu, and T. Li, "Joint device selection and bandwidth allocation for cost-efficient federated learning in industrial Internet of Things," *IEEE Internet Things J.*, vol. 10, no. 10, pp. 9148–9160, May 2023.

[15] K. Zhao et al., "EdgeAdaptor: Online configuration adaption, model selection and resource provisioning for edge DNN inference serving at scale," *IEEE Trans. Mobile Comput.*, vol. 22, no. 10, pp. 5870–5886, Oct. 2023.

[16] J. Ren, J. Sun, H. Tian, W. Ni, G. Nie, and Y. Wang, "Joint resource allocation for efficient federated learning in internet of things supported by edge computing," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, 2021, pp. 1–6.

[17] Y.-D. Lin, Y.-C. Lai, D. Sudyana, and R.-H. Hwang, "Artificial intelligence for internet of things as a service: Small or big data, private or public model, centralized or federated learning?" *Computer*, vol. 56, no. 12, pp. 65–79, Dec. 2023.

[18] T. Guilmeau, E. Chouzenoux, and V. Elvira, "Simulated annealing: A review and a new scheme," in *Proc. IEEE Stat. Signal Process. Workshop (SSP)*, 2021, pp. 101–105.

[19] A. Krizhevsky and G. Hinton, *Learning Multiple Layers of Features from Tiny Images*, Univ. Toronto, Toronto, ON, Canada, 2009.

[20] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.

[21] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2019, pp. 1387–1395.

[22] M.-T. Thai, Y.-D. Lin, Y.-C. Lai, and H.-T. Chien, "Workload and capacity optimization for cloud-edge computing systems with vertical and horizontal offloading," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 1, pp. 227–238, Mar. 2020.

[23] K. Wang et al., "Joint offloading and charge cost minimization in mobile edge computing," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 205–216, 2020.

[24] (Microsoft, Redmond, WA, USA). *Pricing—Bandwidth | Microsoft Azure*. (2023). [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/bandwidth/

[25] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.

Ren-Hung Hwang (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Massachusetts at Amherst. He is the Dean of the College of Artificial Intelligence, National Yang Ming Chiao Tung University (NYCU), Taiwan. Before joining NYCU, he was with National Chung Cheng University, Taiwan, from 1993 to 2022. He served as the Dean of the College of Engineering from 2014 to 2017. He has published more than 250 international journal and conference papers. His current research interests are in deep learning, wireless communications, network security, and cloud/edge/fog computing. He received the IEEE Best Paper Award from IEEE UbiMedia 2018, IEEE SC2 2017, and IEEE IUCC 2014.

Tsai-Ying Chou received the B.S. and M.S. degrees in computer science from National Chung Cheng University, Taiwan, in 2021 and 2023, respectively. She is currently working as an IT Engineer with Taiwan Semiconductor Manufacturing Company Ltd.

Jia-You Lin (Student Member, IEEE) received the master's degree from the Department of Computer Science and Information Engineering, National Chung Cheng University, where she is currently pursuing the Ph.D. degree. Her research interests include deep learning, resource allocation, scheduling optimization, and cloud/edge/fog computing.

Didik Sudyana received the Ph.D. degree from the Department of Electrical Engineering and Computer Science, National Yang Ming Chiao Tung University in 2024, where he is currently a Postdoctoral Fellow with the High-Speed Network Lab. His research interests include cybersecurity, machine learning, and network design and optimization.

Yuan-Cheng Lai received the Ph.D. degree from the Department of Computer and Information Science, National Chiao Tung University in 1997. He joined the faculty of the Department of Information Management, National Taiwan University of Science and Technology in August 2001, where he has been a Distinguished Professor since June 2012. His research interests include performance analysis, software-defined networking, wireless networks, and IoT security.

Ying-Dar Lin (Fellow, IEEE) received the Ph.D. degree in computer science from the University of California at Los Angeles in 1993. He is a Chair Professor of Computer Science with National Yang Ming Chiao Tung University, Taiwan. He was a Visiting Scholar with Cisco Systems, San Jose, from 2007 to 2008, the CEO of Telecom Technology Center, Taiwan, from 2010 to 2011, and the Vice President of National Applied Research Labs, Taiwan, from 2017 to 2018. He cofounded L7 Networks Inc. in 2002 and O'Prueba Inc. in 2018. His research interests include cybersecurity, wireless communications, network softwarization, and machine learning for communications. He has served or is serving on the editorial boards of several IEEE journals and magazines, including the Editor-in-Chief of IEEE COMMUNICATIONS SURVEYS AND TUTORIALS from 2017 to 2020.