DeepRB: Deep Resource Broker Based on Clustered Federated Learning for Edge Video Analytics

Xiaojie Zhang, Saptarshi Debroy, *Senior Member, IEEE*, Peng Wang, and Keqin Li, *Fellow, IEEE*

***Abstract*—Edge computing plays a crucial role in large-scale and real-time video analytics for smart cities, particularly in environments with massive machine-type communications (mMTC) among IoT devices. Due to the dynamic nature of mMTC, one of the main challenges is to achieve energy-efficient resource allocation and service placement in resource—constrained edge computing environments. In this paper, we introduce *DeepRB*, a deep learning-based resource broker framework designed for real-time video analytics in edge-native environments. *DeepRB* employs a two-stage algorithm to address both resource allocation and service placement efficiently. First, it uses a Residual Multilayer Perceptron (*ResMLP*) network to approximate traditional iterative resource allocation policies for IoT devices that frequently transition between active and idle states. Second, for service placement, *DeepRB* leverages a multi-agent federated deep reinforcement learning (DRL) approach that incorporates clustering and knowledge-aware model aggregation. Through extensive simulations, we demonstrate the effectiveness of *DeepRB* in improving schedulability and scalability compared to baseline edge resource management algorithms. Our results highlight the potential of *DeepRB* for optimizing resource allocation and service placement for real-time video analytics in dynamic and resource-constrained edge computing environments.**

***Index Terms*—Energy efficiency, edge computing, resource management, service placement, real-time video analytics.**

I. Introduction

IN RECENT years, driven by the nimbleness and mobility of Internet of Things (IoT) devices, the pervasive adoption of massive Machine-Type Communications (mMTC) [1] has significantly impacted the landscape of mission-critical surveillance scenarios, such as smart city [2] and Industry 4.0 [3] to name a few. Such surveillance scenarios often use video analytics as a technique to automate the analysis of the video content (collected by IoT devices), which usually has strict end-to-end (E2E) processing requirement. Moreover, to obtain high analysis accuracy, most video analytics technologies rely on Artificial Intelligence (AI) models to interpret the video data and extract insights, e.g., object detection and tracking. Considering the low processing speed and battery capacity of embedded computing chips (i.e., CPU, GPU) on modern IoT devices, running such AI models locally on IoT devices although desirable, is far from realistic. On the other hand, streaming large amount of video content to remote cloud datacenter is counterproductive, as it significantly increases overall response time. Therefore, edge computing has emerged as an ideal compromise between IoT device processing and remote cloud-based processing by moving computing resources and data storage to the edge of the network, providing faster, more reliable, and more secure computing services [4].

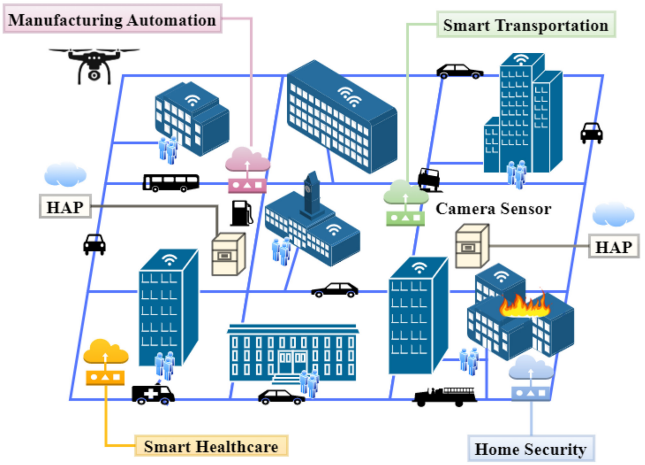


Fig. 1. Use cases of edge-supported video analytics. In this scenario, a large number of IoT devices equipped with camera sensors periodically or occasionally request computing services from nearby Hybrid Access Points (HAPs) [5] to enable real-time and energy-efficient video processing. An HAP typically consists of a standalone Wireless Access Point (WAP) to provide network infrastructure, as well as heterogeneous computing units (e.g., GPUs and CPUs) to provide computation resources.

In Fig. 1, we show an exemplary edge-supported mMTC scenario in the context of a smart city use case, where IoT devices (such as unmanned aerial vehicles, industrial robots, and other monitoring devices) are strategically distributed in an urban area, each serving a specific purpose, such as manufacturing automation [6], smart healthcare [7], smart transportation [8], home security [9], and so on. In such

Received 16 July 2024; revised 6 January 2025 and 2 April 2025; accepted 2 April 2025. Date of publication 15 April 2025; date of current version 7 August 2025. This work is partially supported by Hunan Province Three Point Science and Technology Innovation Plan Youth Science and Technology Talent Project (Grant No. 2022RC1101, PI Wang), and by United States National Science Foundation (Grant No. 1943338, PI Debroy). The associate editor coordinating the review of this article and approving it for publication was P. Calyam. *(Corresponding author: Peng Wang.)*

Xiaojie Zhang and Peng Wang are with the School of Electronics and Information Engineering, and the Key Laboratory of Hunan Province for 3D Scene Visualization and Intelligence Education, Hunan First Normal University, Changsha 410002, China (e-mail: xzhang6@gradcenter.cuny.edu; pwang@hnu.edu.cn).

Saptarshi Debroy is with the Department of Computer Science, City University of New York, New York, NY 10016 USA (e-mail: saptarshi.debroy@hunter.cuny.edu).

Keqin Li is with the College of Information Science and Engineering, and the National Supercomputing Center in Changsha, Hunan University, Changsha 410082, China, and also with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/TNSM.2025.3560657

scenarios, end users (such as private companies, schools, manufacturers, and government entities) utilize IoT devices to capture raw video data of the surrounding environment with targeted scenes or objects. The data is then transmitted to a nearby hybrid access point (HAP) [5]. The video analytics applications hosted by such HAP units process the data (using the computation resources of the HAPs) and feed the results back to the respective consumers. The consumers subsequently use these results for informed decision-making with or without a 'human-in-the-loop'. Such an mMTC ecosystem thus facilitates real-time and context-aware analysis, decision making, and actions that are of paramount importance for the efficient and effective operation of modern smart city use cases [10].

Although edge computing is recognized as a critical enabler for implementing mMTC in dynamic real-world environments, it still faces several challenges. *Firstly*, due to the spatio-temporal variance of the radio spectrum, the wireless channel quality and consequently the radio transmission characteristics experienced by IoT devices in urban areas can vary greatly resulting in *time-varying* network delay, which in turn impacts the end-to-end latency of video analytics. *Secondly*, in smart cities, a vast number of IoT devices that require sporadic or occasional communication [11], [12], are connected to an edge network simultaneously. Such random transitions of IoT devices between active and idle states introduce additional complications to overall system resource management. *Finally*, in edge computing, joint optimization is frequently required for resource allocation and service placement, with discrete and continuous variables tightly interconnected. This need is especially pronounced in dynamic edge environments, where solving long-term Mixed-Integer Nonlinear Programming (MINLP) problems poses a significant challenge.

Works such as [13], [14], [15], [16], [17] use problem decomposition or other similar methods to address above challenges. However, these traditional methods struggle to adapt to the dynamic changes in computing environments and often result in convergence to local optima too easily. In contrast, studies such as [18], [19], [20], [21], [22] leverage deep learning algorithms to address edge dynamism. These approaches empower learning agents to interact within the edge environment, facilitating optimal resource allocation and service placement strategies [23]. Nevertheless, none of the aforementioned work addresse the schedulability and scalability issues in mMTC scenarios, where the number of IoT devices can frequently change. This requires a more flexible model that can adapt to a varying number of IoT devices, a capability that centralized methods lack. To the best of our knowledge, this article is among the few that develop a distributed training framework based on Multi-agent Deep Reinforcement Learning (MADRL), utilizing clustering and knowledge-aware model aggregation inspired by Federated Learning (FL).

In this article, we propose an edge-assisted resource brokering (RB) framework, viz., *DeepRB*, that performs joint resource allocation and service placement for large-scale video analytics. The main contributions are summarized as follows:

1) *System Model:* We study a dynamic scenario where IoT devices may frequently transition between active and idle states, resulting in constant fluctuations in computation and communication demands. We then formulate this fundamental problem as a long-term MINLP problem, aiming to determine energy-efficient resource allocation and service placement in resource-constrained edge computing environments. To reduce the complexity of the proposed problem, we decompose it into two sub-problems: *Resource Allocation* and *Service Placement*.
2) *Resource Allocation:* We design an *Iterative Resource Algorithm* (*IRA*), a robust iterative algorithm based on Lagrangian multipliers and the sub-gradient method to jointly optimize computational and networking resource allocation for enhanced energy efficiency. To boost responsiveness and scalability, we introduce *ResMLP*, a Residual Multi-Layer Perceptron-based model specifically designed to emulate the policies generated by the *IRA*, achieving near-optimal performance with reduced computation time.
3) *Service Placement*: we propose a cutting-edge Proximal Policy Optimization (PPO)-based algorithm to determine optimal service placement strategies, ensuring efficient utilization of distributed resources. To further enhance training efficacy and adaptability, we design a novel Digital Twin (DT)-enabled distributed training framework. This framework incorporates clustering techniques and knowledge-aware Federated Learning (FL), significantly improving the scalability and efficiency of the service placement process.
4) *Validation*: The proposed *DeepRB* framework is extensively validated through a case study using real-world applications, demonstrating its scalability and efficiency in large-scale IoT deployments. The proposed algorithm achieves a 25.8% improvement in task completion ratio compared to naive resource allocation schemes. Additionally, the *ResMLP* model significantly accelerates resource allocation decision-making by a factor of 10X to 500X, nearly without any noticeable loss ($\leq 3.16\%$). For service placement, *DeepRB* shows strong robustness, achieving over 60% energy savings in static environments and a 22% to 25.1% reduction in energy consumption in dynamic scenarios when compared to state-of-the-art methods.

The rest of the paper is organized as follows. Section II presents the related works. Section III proposes the system model and problem formulation. Section IV presents the resource allocation algorithm. Section V discusses the service placement. Section VI discusses performance evaluation. Section VII concludes the paper.

II. Related Works

The performance of real-time video analytics at the edge is heavily influenced by key factors such as resource allocation and service placement strategies—terms also referred to as server assignment or task offloading in related studies. Recently, a growing number of works has focused on achieving energy-efficient video analytics through the joint optimization of these factors. However, this joint optimization is inherently

TABLE I
COMPARISON OF PROPOSED SOLUTION WITH RELATED WORKS

Ref	Stochastic Task	Decomposition	Method
[13]		✓	BCD
[14]		✓	Heuristic
[15]	✓	✓	Lyapunov + Cooperative Game
[16]		✓	Hungarian
[17]		✓	Two-sided Matching Game
[18]	✓		DQN
[19], [20]	✓		Distributed-DDQN
[21]	✓		DDPG
[22]	✓		PPO
Ours	✓	✓	ResMLP + Clustered-PPO

challenging, as it involves solving a complex MINLP problem. Due to its discrete and combinatorial nature, this problem is typically NP-hard [24], posing significant computational and scalability challenges. Table I compares the proposed solution with existing studies in terms of task feature, problem decomposition, and methodology.

A. Problem Decomposition and Iterative-Based Algorithm

Since finding an optimal solution to complex MINLP problems is notably challenging and time-consuming, many existing works focus on designing low-complexity methods (such as decomposition or bi-level optimization, game theory, etc) to obtain near-optimal solutions. In their work [13], the authors introduce *FACT*, a block coordinate descent (BCD) approach, which iteratively tackles the server assignment and frame resolution selection sub-problems by sequentially fixing one variable at a time. In [14], the authors utilize a heuristic algorithm, which iteratively executes either the remove operation or the exchange operation, to derive the binary task offloading decisions. The authors in [15] present *JORA*, a Lyapunov optimization-based online approach, formulating the task offloading problem as a distributed non-cooperative game. Likewise, works such as [16], [17] employ the Hungarian method and a two-sided matching game to devise optimal offloading strategies. However, these works are often required to solve the resource allocation sub-problems repeatedly many times, which is also time-consuming as the number of IoT devices grows; especially for works that rely on iterative parameter update (e.g., gradient descent or sub-gradient descent methods). In addition, none of these works have addressed the aspect of minimizing the decision time for resource allocation. Therefore, they are not designed for dynamic edge computing environments.

B. DRL-Based Algorithm

Due to recent advancements of deep learning technologies, such as Deep Q-Network (DQN) and Double Deep Q-Network (DDQN) [18], [19], DRL-based algorithms have gained significant attention in the domain of edge computing. Compared to the previously mentioned iterative-based algorithms, DRL approaches offer a more efficient and adaptable means of decision-making by allowing continuous interactions between agents and environment. This has led to their popularity in tackling complex and dynamic edge computing problems. Reference [20] proposes *DeepEdge*, a DDQN based algorithm for server selection in a stochastic edge environment with dynamic offloading requests and time-varying communication conditions. The authors in [21] develop a Deep Deterministic Policy Gradient (DDPG) empowered model, called *D3PG*, to optimize task partitioning and resource allocation. In *D3PG*, noises sampled from the Dirichlet distribution are added to actions as the exploration mechanism. Reference [22] presents *EPtask*, which is designed based on Proximal Policy Optimization (PPO) algorithm to make joint decisions on task offloading and priority assignment.

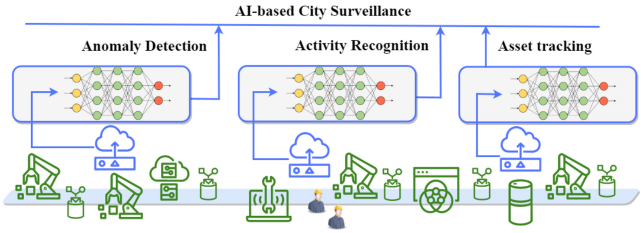


Fig. 2. AI-based city surveillance for smart city through real-time edge IoT video analytics.

Nevertheless, none of the aforementioned works have taken into account the sporadic or occasional communication characteristics of mMTC, combined with a fast and highly adaptable resource allocation policy of HAPs. To the best of our knowledge, our work is the first to address the conjunction of these two aspects.

III. SYSTEM MODEL AND PROBLEM FORMULATION

As depicted in Fig. 2, we consider a standalone edge computing framework tailored for smart city applications, called *DeepRB*, which consists of a set of HAPs $\mathcal{K} = \{1, \ldots, k, \ldots, K\}$ and numerous IoT devices $\mathcal{N} = \{1, \ldots, n, \ldots, N\}$ (it is assumed that $N \gg K$). The IoT devices $n \in \mathcal{N}$ are randomly distributed across the service area, carrying out their designated real-time video analytics missions with the assistance of a selected HAP $k \in \mathcal{K}$ (e.g., anomaly detection, activity recognition, asset tracking). In this article, we particularly focus on IoT devices that operate sporadically or intermittently due to their mission attributes. We define a time-slotted optimization system $\mathcal{T} = \{1, \ldots, t, \ldots, T\}$, where the edge environment is assumed to remain unchanged within each time slot, denoted by t. Moreover, we describe the service placement problem as $a_{n,k}(t) \in \{0, 1\}$, where $a_{n,k}(t) = 1$ indicates that the n-th IoT device is connected to the k-th HAP in time slot t; otherwise, $a_{n,k}(t) = 0$. Since IoT device can be connected to only one HAP, we have $\sum_{k=1}^{K} a_{n,k}(t) = 1, \ \forall n \in \mathcal{N}$ for all $t \in \mathcal{T}$. The *DeepRB* framework is centrally managed by a coordinator, hosted by one or a collection of HAPs acting in unison. It is responsible for orchestrating resource allocation and service placement between HAPs and IoT devices. The major notations used in the system model and problem formulation are summarized in Table II.

TABLE II
TABLE OF NOTATIONS

Symbol	Definition
$\mathcal{K}$	Set of HAPs
$\mathcal{N}$	Set of IoT devices
k	k-th HAP
n	n-th IoT device
$a_{n,k}(t)$	Service placement indicator between IoT device n and HAP k
(x_n, y_n)	2-D coordinate of n-th IoT device
v_n	Activation state of n-th IoT device
τ	Latency requirement of IoT devices
$t^C_{n,k}(t)$	Task computation latency of n-th IoT device on k-th HAP
m_n	DNN model type of task for n-th IoT device
β_n	Number of captured frames per time slot for n-th IoT device
u_n	Resolution of captured frames for n-th IoT device
$C_{m_n}(u_n)$	Computational complexity of the task for n-th IoT device with DNN model m_n and resolution u_n
$f_{n,k}(t)$	Allocated computational resources from k-th HAP to the n-th IoT device
F^{max}_k	Computation capacity of k-th HAP
$w_{n,k}(t)$	Allocated bandwidth from k-th HAP to the n-th IoT device
W_k	Available bandwidth of k-th HAP
$r_{n,k}(t)$	Available data rate for n-th IoT device when connected to k-th HAP
$p_{n,k}(t)$	Transmission power for IoT device n when connected to k-th HAP
$g_{n,k}$	Channel power gain for IoT device n when connected to k-th HAP
$T^D_{n,k}(t)$	Task transmission latency for IoT device n when connected to k-th HAP
$D(u_n)$	Data size of tasks released by n-th IoT device with frame resolution u_n
$e_{n,k}(t)$	Energy consumption of n-th IoT device when connected to k-th HAP
P^{max}	Maximum transmission power of IoT device

A. IoT Device State and Application Model

As previously mentioned, IoT devices are distributed randomly across the service area. We define $loc_n = (x_n, y_n)$ and $loc'_k = (x'_k, y'_k)$ as the geolocation of *n*-th IoT device and *k*-th HAP, respectively. Then, the euclidean distance between the *n*-th IoT device and *k*-th HAP is defined as $\psi_{n,k} = ||loc_n - loc'_k||$. It is assumed that each IoT device may transition to idle state in each time slot with probability ρ. We set $v_n(t) = 1$ if the IoT device *n* is in active state; otherwise, $v_n(t) = 0$. Moreover, our primary focus is on Deep Neural Networks (DNNs) for video analytics, which are extensively utilized in public safety and automation scenarios. Specifically, we concentrate on DNN models that accept individual video frames as inputs and generate customized results accordingly.

In this article, we define the features of computation tasks as a tuple $\varsigma_n = \{\beta_n, m_n, u_n\}$, denoting the number of captured frames per time slot β_n, the application's DNN model $m_n \in \mathcal{M} = \{1, \ldots, m, \ldots, M\}$ (e.g., *yolov5, SSD, AlexNet*), and the frame resolution $u_n \in \mathcal{U} = \{128 \times 128, \ldots, 640 \times 640\}$, respectively. We assume that IoT devices continuously generate such tasks, and each task must be processed within a deadline τ. We recognize that task heterogeneity poses a significant challenge, especially as the number of active IoT devices grows. Thus, jointly optimizing resource allocation and service placement becomes critical for the efficient operation of the *DeepRB* framework.

B. Analytical Models

1) Computation Model: Define the computational complexity of a task as $\beta_n \times C_{m_n}(u_n)$, the computation latency is computed by

$$t^C_{n,k}(t) = \begin{cases} \frac{\beta_n \times C_{m_n}(u_n)}{f_{n,k}(t)} & a_{n,k}(t)v_n(t) = 1 \\ 0 & a_{n,k}(t)v_n(t) = 0 \end{cases} \quad (1)$$

where $C_{m_n}(u_n)$ (measured in CPU cycles) is the computational requirements of a single inference on DNN model m_n with frame resolution u_n and $f_{n,k}(t)$ is the amount of computational resources allocated from *k*-th HAP to the *n*-th IoT device in time slot *t*. It needs to satisfy the following constraint for all $t \in \mathcal{T}$:

$$\sum_{n=1}^{N} f_{n,k}(t) = F^{max}_k, \ \forall k \in \mathcal{K} \quad (2)$$

where F^{max}_k is the computation capacity of *k*-th HAP.

2) Transmission Model: We consider Orthogonal Frequency-Division Multiple Access (OFDMA) [25] based communication networks, where each HAP occupies an exclusive wireless channel whose bandwidth is denoted as W_k, $\forall k \in \mathcal{K}$ (measured in Hz). Additionally, we define $\mathbf{w}_k(t) = [w_{1,k}(t), \ldots, w_{n,k}(t), \ldots, w_{N,k}(t)]$ as the bandwidth allocation vector of the *k*-th HAP, where $w_{n,k}(t)$ represents the bandwidth of a sub-channel allocated to IoT device *n*. Our model does not account for either inter-HAP or intra-HAP interference, therefore the follow constraint needs to be satisfied

$$\sum_{n=1}^{N} w_{n,k}(t) = W_k, \ \forall k \in \mathcal{K} \quad (3)$$

Given $\mathbf{w}_k(t)$, the available data rate experienced by IoT device *n* connected to the *k*-th HAP is expressed by the Shannon-Hartley formula as follows:

$$r_{n,k}(t) = \begin{cases} w_{n,k}(t)\log_2\left(1 + \frac{p_{n,k}(t)g_{n,k}}{N_0}\right) & a_{n,k}(t)v_n(t) = 1 \\ 0 & a_{n,k}(t)v_n(t) = 0 \end{cases} \quad (4)$$

where $p_{n,k}(t) \leq P^{max}$ is the transmission power of IoT device *n*, $g_{n,k} = g(\psi_{n,k})$ is the channel power gain between IoT device *n* and HAP *k* (we consider a log-distance path loss mode, therefore the channel gain is a function of distance), and N_0 indicates the constant background noise variance [26], [27], [28]. To estimate the data size of a video frame, we adopt the model used in [29] and define data size as $D(u_n) = \sigma u_n^2$ (measured in bits), where σ is a constant.

Thus, the transmission latency is computed by

$$t^D_{n,k}(t) = \begin{cases} \beta_n \times \frac{D(u_n)}{r_{n,k}(t)} & a_{n,k}(t)v_n(t) = 1 \\ 0 & a_{n,k}(t)v_n(t) = 0 \end{cases} \quad (5)$$

3) Energy Model: In this article, the energy consumption of IoT devices refers to the energy used during the data offloading phase, which is computed by

$$e_{n,k}(t) = \begin{cases} t^D_{n,k}(t)p_{n,k}(t) & a_{n,k}(t)v_n(t) = 1 \\ 0 & a_{n,k}(t)v_n(t) = 0 \end{cases} \quad (6)$$

C. Problem Formulation

DeepRB aims to achieve energy-efficient resource allocation and service placement in resource-constrained edge computing environments. First, let us define the following sets of variables that need to be optimized by *DeepRB*:

1) The transmission power of IoT devices $\mathbf{p}_k(t) = [p_{1,k}(t), \ldots, p_{n,k}(t), \ldots, p_{N,k}(t)]$ for all $k \in \mathcal{K}$.
2) The service placement vector of IoT devices $\mathbf{a}_k(t) = [a_{1,k}(t), \ldots, a_{n,k}(t), \ldots, a_{N,k}(t)]$ for all $k \in \mathcal{K}$.
3) The computational resource allocation vector $\mathbf{f}_k(t) = [f_{1,k}(t), \ldots, f_{n,k}(t), \ldots, f_{N,k}(t)]$ for every $k \in \mathcal{K}$.
4) The network resource allocation vector $\mathbf{w}_k(t) = [w_{1,k}(t), \ldots, w_{n,k}(t), \ldots, w_{N,k}(t)]$ for every $k \in \mathcal{K}$.

In *DeepRB*, the proposed long-term optimization problem is stated as

$$\min_{\{\mathbf{p},\mathbf{a},\mathbf{f},\mathbf{w}\}} \quad \lim_{T\to+\infty} \frac{1}{T}\sum_{t=1}^{T}\left(\sum_{n=1}^{N}\sum_{k=1}^{K} e_{n,k}(t)\right)$$

$$\text{s.t.} \quad \mathbf{C1}: \sum_{n=1}^{N} f_{n,k}(t) = F_k^{max}, \ \forall k \in \mathcal{K}, \ \forall t \in \mathcal{T}$$

$$\mathbf{C2}: \sum_{n=1}^{N} w_{n,k}(t) = W_k, \ \forall k \in \mathcal{K}, \ \forall t \in \mathcal{T}$$

$$\mathbf{C3}: \sum_{k=1}^{K} a_{n,k}(t) = 1, \ \forall n \in \mathcal{N}, \ \forall t \in \mathcal{T}$$

$$\mathbf{C4}: p_{n,k}(t) \le P^{max}, \ \forall n \in \mathcal{N}, \ \forall k \in \mathcal{K}, \ \forall t \in \mathcal{T}$$

$$\mathbf{C5}: \sum_{k=1}^{K}\left(t^D_{n,k}(t) + t^C_{n,k}(t)\right) \le \tau, \ \forall n \in \mathcal{N}, \ \forall t \in \mathcal{T}$$

$$\mathbf{C6}: a_{n,k}(t) \in \{0,1\}, \ \forall n \in \mathcal{N}, \ \forall k \in \mathcal{K}, \ \forall t \in \mathcal{T} \quad \text{(P1)}$$

In (P1), **C1** and **C2** specify the computational and network resource allocation constraints. **C3** indicates that IoT devices can only be connected to one HAP in any time slot. **C4** states the range of transmission power of IoT devices. Finally, **C5** presents the task deadline requirement of IoT devices.

We notice that Problem (P1) is a classic MINLP problem, solving such type of problems in a long-term manner is notoriously challenging mainly due to following reasons: firstly, the coupling of resource allocation and service placement results in a combinatorial optimization problem. Secondly, the state of IoT devices is a time-varying parameter, requiring that Problem (P1) be solved repeatedly and continuously. However, with the number of IoT devices increases, while traditional iterative-based optimization methods may find high-quality solutions, they can be very time-consuming. Therefore, the goal of *DeepRB* is to solve Problem (P1) rapidly and efficiently.

D. Solution Methodology

In this article, to address above challenges, we apply problem decomposition to convert Problem (P1) into following two sub-problems:

1) *Resource Allocation* (RA) - Section IV: Given service placement vectors $\mathbf{a}_1(t), \ldots, \mathbf{a}_k(t), \ldots, \mathbf{a}_K(t)$, Problem (P1) is decomposed into multiple identical resource allocation sub-problems, denoted by $\mathcal{P}_1(t), \ldots, \mathcal{P}_k(t), \ldots, \mathcal{P}_K(t)$ (dedicated to each HAP). Each sub-problem $\mathcal{P}_k(t)$ only serves the connected IoT devices, which determines resource allocation vectors $\mathbf{p}_k(t)$, $\mathbf{f}_k(t)$ and $\mathbf{w}_k(t)$.
2) *Service Placement* (SP) - Section V: Once the resource allocation policy is obtained, Problem (P1) becomes a classic zero-one integer linear programming problem. The service placement sub-problem is denoted by $\mathcal{P}_{sp}(t)$, which determines service placement vectors $\mathbf{a}_1(t), \ldots, \mathbf{a}_k(t), \ldots, \mathbf{a}_K(t)$.

The interrelationship between the two sub-problems is depicted in Fig. 3. In Section IV, we address the first sub-problem (RA). We first introduce an iterative resource allocation algorithm as the baseline solution, which uses the subgradient update method. Subsequently, to enable rapid decision-making, we design a residual MLP network to approximate the policy of the proposed iterative algorithm. In Section V, the second sub-problem (SP) is solved repeatedly and continuously using a novel distributed DRL algorithm, where the learning agents are clustered. In each iteration, the solution to the first sub-problem is utilized by the DRL algorithm to evaluate the quality of the actions it selects for service placement.

IV. MLP-Based Joint Network and Computational Resource Allocation

We first define $N^{max} \in \mathbb{Z}^+$ as the maximum number of IoT devices allowed to connect to each HAP (pre-defined from HAP's experience). With fixed service placement vectors $\mathbf{a}_1(t), \ldots, \mathbf{a}_k(t), \ldots, \mathbf{a}_K(t)$, we use $\mathcal{N}_k(t) = \{n \mid a_{n,k}(t) = 1, \ ; \forall n \in \mathcal{N}\}$ to represent the subset of IoT devices that are connected to the k-th HAP ($|\mathcal{N}_k(t)| \le N^{max}$). In this section, the resource allocation sub-problem $\mathcal{P}_k(t)$ which works with $\mathcal{N}_k(t)$ and HAP k, is stated as

$$\mathcal{P}_k(t) \triangleq \min_{\{\mathbf{f}_k(t),\mathbf{w}_k(t),\mathbf{p}_k(t)\}} \quad \sum_{n\in\mathcal{N}_k(t)} e_{n,k}(t)$$

$$\text{s.t.} \quad \mathbf{C1}: \sum_{n\in\mathcal{N}_k(t)} f_{n,k}(t) = F_k^{max}$$

$$\mathbf{C2}: \sum_{n\in\mathcal{N}_k(t)} w_{n,k}(t) = W_k$$

$$\mathbf{C3}: t^D_{n,k}(t) + t^C_{n,k}(t) \le \tau, \ \forall n \in \mathcal{N}_k(t) \quad \text{(P2)}$$

Obviously, we need to solve $\mathcal{P}_k(t) \triangleq$ Problem (P2) for individual HAPs. Nevertheless, Problem (P2) is still time-consuming to solve due to the coupling of $\mathbf{f}_k(t)$ and $\mathbf{w}_k(t)$ in constraint **C3** among all IoT devices in $\mathcal{N}_k(t)$.

To solve Problem (P2), we first design an iterative algorithm (called *IRA*) by utilizing Lagrangian multipliers and a subgradient method. Moreover, we propose a residual MLP network (called *ResMLP*) to estimate the outcome of *IRA* algorithm, aiming to significantly shorten the decision time. The proposed *ResMLP* network is a crucial enhancement for incorporating all resource allocation sub-problems $\{\mathcal{P}_1(t), \ldots, \mathcal{P}_k(t), \ldots, \mathcal{P}_K(t)\}$ into the service placement $\mathcal{P}_{sp}(t)$, as its fast decision-making speed enables the DRL agent to be trained online (this will be introduced in the next section).

A. Iterative-Based Algorithm for Resource Allocation

In this subsection, let us first define the Lagrange function of Problem (P2), which is stated as follows

$$\begin{aligned}\mathrm{L}_{\mathcal{P}_k(t)}(\mathbf{f}_k(t), \mathbf{w}_k(t), \mathbf{p}_k(t), \boldsymbol{\omega}) = &\sum_{n\in\mathcal{N}_k(t)} e_{n,k}(t) \\ &+ \sum_{n\in\mathcal{N}_k(t)} \mu_{n,k}\left(t^D_{n,k}(t) + t^C_{n,k}(t) - \tau\right) \\ &+\lambda_k\left(\sum_{n\in\mathcal{N}_k(t)} f_{n,k}(t) - F^{max}_k\right) + \nu_k\left(\sum_{n\in\mathcal{N}_k(t)} w_{n,k}(t) - W_k\right)\end{aligned} \tag{7}$$

where $\boldsymbol{\omega}$ are sets of Lagrangian multipliers, including λ_k, ν_k and $\boldsymbol{\mu}_k = [\mu_{1,k}, \ldots, \mu_{n,k}, \ldots, \mu_{N,k}]$. These multipliers are associated with constraints **C1**, **C2** and **C3**, respectively. To make *DeepRB* agile to the time-varying IoT device states, *DeepRB* further reduces the complexity of $\mathcal{P}_k(t)$ by applying a 2-step method.

1) STEP 1 – Optimal Resource Allocation $\mathbf{f}_k(t)$ and $\mathbf{w}_k(t)$: In 1^{st} step, we solve Problem (P2) by fixing $\mathbf{p}_k(t)$. By fixing $\mathbf{p}_k(t)$, we can easily observe that the elements in Hessian matrix of Eq. (7) are as follows

$$\frac{\nabla^2 \mathrm{L}_{\mathcal{P}_k(t)}}{\nabla f_{n,k}(t)\, \nabla\, f_{n',k}(t)} = \begin{cases} \mu_{n,k}\beta_n \times \frac{2C_{m_n}(u_n)}{\left[f_{n,k}(t)\right]^3} & n = n' \\ 0 & n \neq n' \end{cases} \tag{8}$$

and

$$\frac{\nabla^2 \mathrm{L}_{\mathcal{P}_k(t)}}{\nabla w_{n,k}(t)\, \nabla\, w_{n',k}(t)} = \begin{cases} \frac{(p_{n,k}(t)+\mu_{n,k})\beta_n}{r_{n,k}(t)} \times \frac{2D(u_n)}{\left[w_{n,k}(t)\right]^3} & n = n' \\ 0 & n \neq n' \end{cases} \tag{9}$$

which are positive semi-definite in terms of variables $\mathbf{f}_k(t)$ and $\mathbf{w}_k(t)$, making Problem (P2) a convex problem. Since Problem (P2) is convex and it satisfies the Slater's condition [30], Problem (P2) holds strong duality (by fixing $\mathbf{p}_k(t)$). Thus, Problem (P2) can be solved by alternatively solving its dual problem. As the Lagrange function of Problem (P2) is differentiable, we apply a sub-gradient method to find the optimal solution by updating the multipliers iteratively.

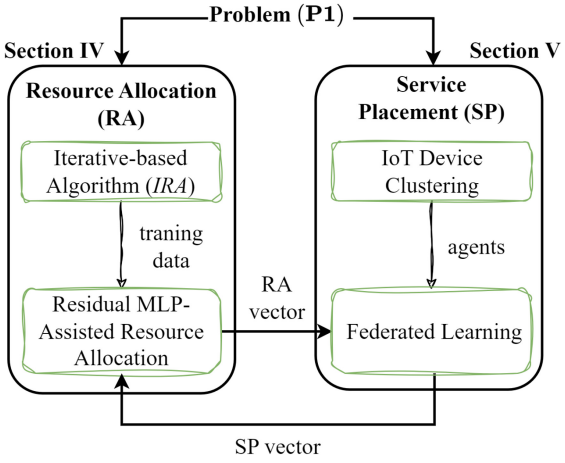


Fig. 3. Problem Solving Methodology.

Under given $\boldsymbol{\omega}$, let the derivation of Eq. (7) *w. r. t.* variables $\mathbf{f}_k(t)$ and $\mathbf{w}_k(t)$ be zero, i.e.,

$$\frac{\nabla \mathrm{L}_{\mathcal{P}_k(t)}}{\nabla f_{n,k}(t)} = 0, \frac{\nabla \mathrm{L}_{\mathcal{P}_k(t)}}{\nabla w_{n,k}(t)} = 0 \tag{10}$$

By solving Eq. (10), the optimal solution $\mathbf{f}^*_k(t)$ and $\mathbf{w}^*_k(t)$ are obtained either though closed-form expression or bisection search method.

2) STEP 2 – Optimal Transmission Power $\mathbf{p}_k(t)$: In 2^{nd} step, we solve Problem (P2) by conversely fixing $\mathbf{f}_k(t)$ and $\mathbf{w}_k(t)$ (which are obtained from the 1^{st} step). We then define the following helper function

$$\mathbf{g}\left(r_{n,k}(t)\right) = N_0\left(2^{\frac{r_{n,k}(t)}{w_{n,k}(t)}} - 1\right) \tag{11}$$

which is a monotonically increasing function $w.\ r.\ t.$ data rate $r_{n,k}(t)$. Since $r_{n,k}(t) = \beta_n \times \frac{D(u_n)}{t^D_{n,k}(t)}$, the transmission power can alternatively be expressed as:

$$p_{n,k}(t) = \mathbf{min}\left\{\frac{\mathbf{g}\left(r_{n,k}(t)\right)}{g\left(\psi_{n,k}\right)}, P^{max}\right\} \tag{12}$$

When $\Delta_{n,k}(t) = \tau - (t^D_{n,k}(t) + t^C_{n,k}(t)) > 0$, we allow IoT devices to adjust their transmission time $t^D_{n,k}(t)$ with the purpose of energy saving and making $\Delta_{n,k}(t) = 0$. The Problem (P3) of finding the optimal transmission time is defined as follows

$$\begin{aligned}\min_{\{t^D_{n,k}(t),\, \forall n\in\mathcal{N}_k(t)\}} \quad & \sum_{n\in\mathcal{N}_k(t)} t^D_{n,k}(t) \times \frac{\mathbf{g}\left(r_{n,k}(t)\right)}{g\left(\psi_{n,k}\right)} \\ \text{s.t. } \mathbf{C1}: \quad & 0 < t^D_{n,k}(t) \leq \tau - t^C_{n,k}(t),\ \forall n \in \mathcal{N}_k(t) \\ \mathbf{C2}: \quad & \frac{\mathbf{g}\left(r_{n,k}(t)\right)}{g\left(\psi_{n,k}\right)} \leq P^{max},\ \forall n \in \mathcal{N}_k(t)\end{aligned} \tag{P3}$$

Considering that $\mathbf{g}''(r_{n,k}(t)) > 0$, objective function $t^D_{n,k}(t) \times \frac{\mathbf{g}(r_{n,k}(t))}{g(\psi_{n,k})}$ is convex in terms of $t^D_{n,k}(t)$ [31], thus Problem (P3) is convex. The optimal solution is stated as

$$t^{D,*}_{n,k}(t) = \frac{\beta_n D(u_n) ln(2)}{w_{n,k}(t)\left[W_0\left(\frac{-\mu_{n,k} g(\psi_{n,k}) + N_0}{-N_0 e}\right) + 1\right]} \tag{13}$$

Algorithm 1: Iterative Resource Allocation (*IRA*)

Initialization: input $k \in \mathcal{K}$, $\mathcal{N}$, $\mathbf{a}_k(t)$, and $\boldsymbol{\omega}$.
Set $\mathcal{N}_k(t) = \{n \mid a_{n,k}(t) = 1,\ \forall n \in \mathcal{N}\}$.
Set $\mathbf{f}_k(t)$ and $\mathbf{w}_k(t)$ with *FAIR* resource allocation policy.
Set $\mathbf{p}_k(t)$ with P^{max}.
while $j \leq j_{max}$ **do**
  # Update variables:
  for *each* $n \in \mathcal{N}_k(t)$ **do**
    Obtain $f_{n,k}(t)$ and $w_{n,k}(t)$ by solving Eq. (10)
    Compute $t^D_{n,k}(t)$ based on Eq. (13)
    Obtain $p_{n,k}(t)$ from Eq. (12)
  # Update multipliers:
  $\mu^{j+1}_{n,k} = \left[\mu^j_{n,k} + \Delta(j)(t^D_{n,k}(t) + t^C_{n,k}(t) - \tau)\right]^+$
  $\lambda^{j+1}_k = \left[\lambda^j_k + \Delta(j)(\sum_{n\in\mathcal{N}_k} f_{n,k}(t) - F^{max}_k)\right]^+$
  $\nu^{j+1}_k = \left[\nu^j_k + \Delta(j)(\sum_{n\in\mathcal{N}_k} w_{n,k}(t) - W_k)\right]^+$
  if $|t^D_{n,k}(t) + t^C_{n,k}(t) - \tau| \leq \epsilon,\ \forall n \in \mathcal{N}_k(t)$ **then**
    break
  j++
return $\{\mathbf{f}_k(t), \mathbf{w}_k(t), \mathbf{p}_k(t)\}$

$s.t., t^{D}_{n,k}{}^{*}(t) \leq \frac{\beta_n D(u_n)}{r^{max}_{n,k}(t)}$, where $r^{max}_{n,k}(t)$ is the maximum data rate when set $p_{n,k}(t) = P^{max}$. As $w_{n,k}(t)$ and $D(u_n)$ are already given, the transmission power $p_{n,k}(t)$ can be computed by Eq. (12) once Problem (P3) is solved.

3) Subgradient Method: The ultimate solution (i.e., $\mathbf{f}_k(t), \mathbf{w}_k(t), \mathbf{p}_k(t)$) of Problem (P2) is derived by performing the above two steps alternatively and repeatedly, with gradually updating the Lagrangian multipliers $\boldsymbol{\omega}$. The proposed iterative-based algorithm, called *IRA*, is depicted in Algorithm 1. In *IRA*, $\mathbf{f}_k(t)$ and $\mathbf{w}_k(t)$ are initialized with *FAIR* resource allocation policy, where resources are evenly distributed to connected IoT devices $\mathcal{N}_k(t)$. Other settings are as follows. $\Delta(j)$ indicates the diminishing step size (e.g., $\{1/j\}^{\frac{1}{2}}$) and j_{max} is maximum number of steps allowed for updates. Specifically, any feasible set of $\boldsymbol{\omega}$ which leads to $|t^D_{n,k}(t) + t^C_{n,k}(t) - \tau| \leq \epsilon,\ \forall n \in \mathcal{N}_k(t)$ yields an optimal solution to Problem (P2). ϵ is the maximum tolerance of task deadline (e.g., 10^{-2} sec).

B. Residual MLP-Assisted Resource Allocation

However, Problem (P1) is a long-term optimization problem which requires us to solve multiple sub-problems (P2) $\triangleq \{\mathcal{P}_1(t), \ldots, \mathcal{P}_k(t), \ldots, \mathcal{P}_K(t)\}$ repeatedly using Algorithm 1. This can be problematic, especially for edge environments with a large number of IoT devices. Beside, the state of IoT devices is a time-varying parameter (active $v_n(t) = 1$ or idle $v_n(t) = 0$), each HAP is going to serve a distinct subset of IoT devices in different time slots, adding an additional layer of complexity. Obviously, the *IRA* algorithm in Algorithm 1 can not satisfy these requirements.

Our design goal is to make resource allocation decisions on the fly for arbitrary set of IoT devices with heterogeneous attributes. The decision time should be measured in a few milliseconds and it should not increase with the size of $\mathcal{N}_k(t)$. In *DeepRB*, we design a Residual Multilayer Perceptron (MLP) network, called *ResMLP*, to approximate the *IRA* algorithm, which is stated as

$$\pi(\mathcal{N}_k(t); \theta) \rightarrow \{\mathbf{f}_k(t),\ \mathbf{w}_k(t)\},\ \forall \mathcal{N}_k(t) \subseteq \mathcal{N},\ \forall k \in \mathcal{K} \tag{14}$$

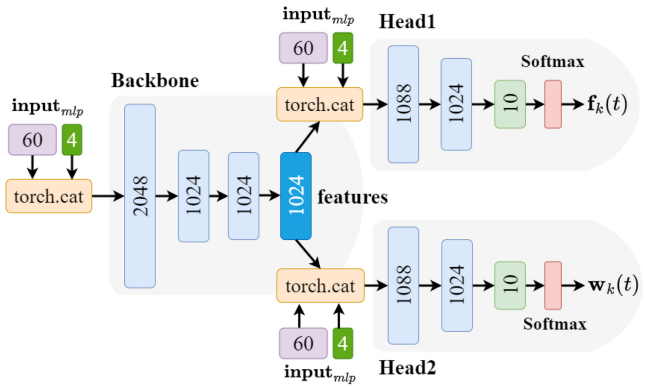


Fig. 4. An example of *ResMLP* based resource allocation, where $N^{max} = 10$. The numbers represent the parameter size of each fully connected layer.

Once $\mathbf{f}_k(t)$ and $\mathbf{w}_k(t)$ are given, the optimal transmission power $\mathbf{p}_k(t)$ can be computed by Eq. (12) and Eq. (13). As shown in Fig. 4, the *ResMLP* structure consists of a backbone for feature extraction and two heads dedicated to resource allocation.

1) Input to ResMLP: The input to *ResMLP* concatenates a vector containing multiple pieces of IoT attributes with a vector representing specific HAP information, i.e.,

$$\mathbf{input}_{mlp} = cat\big(\big[\mathbf{input}_{iot}, \mathbf{input}_{hap}\big]\big)$$

- $\mathbf{input}_{iot}$ contains N^{max} sub-sequences, with the sub-sequence ID be denoted by $i \in [1, N^{max}]$. Each sub-sequence stores the attributes of a particular IoT device $n \in \mathcal{N}$, including the location of this IoT device $(x_n(t), y_n(t))$, its distance to the k-th HAP $\psi_{n,k}$, its DNN model m_n, frame resolution u_n and frame rate β_n, respectively. By assigning specific sub-sequences as all zeros, HAP regulates the number of connected IoT devices.
- $\mathbf{input}_{hap}$ contains the location of k-th HAP $(x'_k(t), y'_k(t))$, its computational capacity F^{max}_k and total bandwidth W_k.

Before moving into the two heads, *ResMLP* concatenates the feature vector generated by the backbone with its original input $\mathbf{input}_{mlp}$. This skip connection helps in learning more complex feature representations, thereby enhancing the network's expressive power. Additionally, it effectively reduces gradient attenuation during network propagation and accelerates the training process.

2) Output of ResMLP: The output of *ResMLP* includes two heads, one for $\mathbf{f}_k(t)$ and the other for $\mathbf{w}_k(t)$, both of which are normalized. To satisfy the resource allocation constraints outlined in Problem (P2) (i.e., **C1** and **C2**), and to ensure that the sum of the head outputs precisely equals to 1 (normalized), a *softmax* layer is appended to each output layer as an activation function.

3) Data Augmentation: DeepRB implements a data augmentation scheme by randomly distributing the attribute sequences from all connected IoT devices $\mathcal{N}_k(t)$ across different sub-sequences within $\mathbf{input}_{iot}$. In other words, the mapping between sub-sequence ID *i* and IoT device ID *n* is arbitrary. This strategy not only increases the training data size but also enhances the generalization ability of *ResMLP*, thereby optimizing training speed and improving accuracy.

V. DRL-Based Distributed Service Placement

In this section, we aim to address sub-problem $\mathcal{P}_{sp}(t)$ for service placement. Here, $\mathcal{P}_{sp}(t)$ becomes a classic 0-1 integer linear programming problem. To solve $\mathcal{P}_{sp}(t)$, we can relax the binary variables $a_{n,k}(t) \in \{0,1\}$ to $\hat{a}_{n,k}(t) \in [0,1]$, and $\hat{a}_{n,k}(t)$ can be interpreted as the fraction of time that the *n*-th IoT device utilizes the resources on the *k*-th HAP.

Although there are well-studied methods that can solve this relaxed problem, most of them require a huge number of iterations (just like Algorithm 1), which makes them impractical and prone to converge to local optima. As the number of IoT devices increases, their effectiveness in addressing $\mathcal{P}_{sp}(t)$ diminishes due to the necessity for rapid decision-making in response to the frequent and ongoing changes in the states of IoT devices. In this paper, we propose a DRL-based distributed learning algorithm, where agents are trained collaboratively facilitated by knowledge transfer and clustered model aggregation.

A. Markov Decision Process

Compared to traditional mathematical methods, reinforcement learning (RL) allows the system to learn from the interactions between agents and the environment. Such unsupervised learning strategy yields an online algorithm that is more adaptable to long-term optimization. Let us consider a typical Markov Decision Process (MDP) for service placement $\mathcal{P}_{sp}(t)$, i.e., a tuple $(\mathcal{S}, \mathcal{A}, \chi, \mathrm{R}, \gamma)$ where $\mathcal{S}$ and $\mathcal{A}$ are the state space and action space, respectively. More specifically, at time *t*, an agent observes a state $s_n(t) \in \mathcal{S}$ and takes an action $a_n(t) \in \mathcal{A}$. Then, based on state transition probability (a matrix), its environment state changes from $s_n(t)$ to $s_n(t+1)$ with probability $\chi(s_n(t+1)|s_n(t), a_n(t))$. Subsequently, the agent is rewarded with a real-value $\mathrm{R}(s_n(t), a_n(t)) \in \mathbb{R}$ that shows the agent's short-term payoff for being in state $s_n(t)$ and taking action $a_n(t)$. The objective is to maximize the expected cumulative discounted reward, which is stated as follows:

$$J_n(\pi) = \mathbb{E}\left[\sum_{t=1}^{+\infty} \gamma^t \mathrm{R}(s_n(t), \pi(s_n(t);\theta))|s_n(1)\right] \quad (15)$$

where $\gamma \in (0,1)$ is the discount factor. The discount factor ensures that the expected cumulative reward is bounded and enables the agent to focus on long-term goals.

1) State: In *DeepRB*, the state of IoT devices, i.e., $s_n(t) \in \mathcal{S}$ specifies the observable environmental and system parameters, which is defined as:

$$\begin{aligned} s_n(t) = \{ & \\ & \{a_{n,1}(t), \ldots, a_{n,K}(t)\} \; //Serviceplacementresults \\ & \{F_1^{max}, \ldots, F_K^{max}\} \; //ComputationresourcesofHAPs \\ & \{W_1, \ldots, W_K\} \; //NetworkresourcesofHAPs \\ & \{(x_1', y_1'), \ldots, (x_K', y_K')\} \; //LocationsofHAPs \\ & \{p_{n,k}(t), \ldots, p_{n,K}(t)\} \\ & //TransmissionPowersofIoTdevice \\ & \{(x_n, y_n), m_n, u_n, \beta_n\} \; //Attributesofn-thIoTdeivce \\ & \{\sum_{k=1}^{K} e_{n,k}(t), \sum_{k=1}^{K} \left(t_{n,k}^{D}(t) + t_{n,k}^{C}(t)\right)\} \\ & //Energyconsumptionandlatency \\ & done_n(t) \; //Systemindicator \\ & \} \end{aligned} \quad (16)$$

We set $done_n(t) = 1$ if the IoT device can finish its tasks within deadline τ; otherwise $done_n(t) = 0$.

2) Action: The action $a_n(t) \in \mathcal{A}$ that an IoT device can take is the selection of HAPs, expressed as:

$$a_n(t) = \{\hat{a}_{n,1}(t), \ldots, \hat{a}_{n,K}(t)\} \quad (17)$$

where $\hat{a}_{n,k}(t)$ is a relaxed value and is converted back to integer value by

$$a_{n,k}(t) = \begin{cases} 1, & k = argmax_{k \in \mathcal{K}} \hat{a}_{n,k}(t) \\ 0, & otherwise \end{cases} \quad (18)$$

3) Reward: In *DeepRB*, we let all IoT devices obtain a shared reward and we design a partial reward function, which is formulated as:

$$\begin{aligned} \mathrm{R}(s_n(t), a_n(t)) = & \frac{\sum_{n=1}^{N} done_n(t)}{\sum_{n=1}^{N} v_n(t)} \\ & \times \frac{1}{\sum_{n=1}^{N}\sum_{k=1}^{K} e_{n,k}(t)} \end{aligned} \quad (19)$$

The objectives embedded within Eq. (19) are twofold: Firstly, it aims to ensure that the ratio $\sum_{n=1}^{N} done_n(t) / \sum_{n=1}^{N} v_n(t) = 1$, indicating successful completion of tasks by all active IoT devices within the given deadline. Secondly, it seeks to minimize the overall energy consumption. In contrast with deterministic reward function, which only assigns a positive reward when $\sum_{n=1}^{N} done_n(t) / \sum_{n=1}^{N} v_n(t) = 1$, the advantage of employing such partial reward function lies in its ability to gradually guide the search towards feasible solutions, particularly beneficial when the pool of feasible solutions is limited.

B. Issues of Distributed Training Data

Many DRL methods outlined in Section II rely on experience replay technique to improve learning efficiency and stability. In particular, an experience is denoted as a tuple $(s_n(t), a_n(t), \mathrm{R}(s_n(t), a_n(t)), s_n(t+1))$, which is stored into a replay memory $\mathcal{D}_n$, $\forall n \in \mathcal{N}$. With fixed intervals, the agents are trained by a random mini-batch from $\mathcal{D}_n$ (while some DRL agents utilize the entire $\mathcal{D}_n$ and subsequently clear it). However, due to the heterogeneous attributes of IoT devices,

$\mathcal{D}_n$ can be highly unbalanced, which may lead to sub-optimal solution due to the absence of coordination among IoT devices (lacking a global view). Moreover, IoT devices that frequently switch to idle state or recently enter the service area may show limited learning ability of their agents. This situation could potentially result in unexpected performance disruptions for other IoT devices. Clearly, it calls for coordinated model training among IoT devices.

C. DT-Empowered DRL Agents

On the other hand, IoT devices are generally subjecting to limited computational capability and low energy budget, making it impractical for local training. Therefore, as illustrated in Fig. 5, we introduce the concept of Digital Twin (DT) system, in which *DeepRB* can be implemented. Through real-time monitoring and emulation of their physical counterparts using specialized IoT communication protocols (e.g., Zigbee, Bluetooth, MQTT), the twin objects within the DT system can easily establish the modeling of accuracy, latency and energy consumption (that were described in Section III-B). With these analytic models, IoT devices operating under computational constraints can effectively host and train their DRL agents within the DT system using twin objects. Additionally, DT simulations can serve as a valuable tool for generating abundant and lifelike training data tailored for the proposed *ResMLP* model (proposed in Section IV-B). However, *DeepRB* does not focus on constructing a DT system, as that is beyond the scope of this article. Instead, we presume the existence of such a DT system maintained by multiple HAPs, as proposed by many state-of-the-art system models [32], [33], [34], [35]. Without loss of generality, we refer to the DRL agents operating within the DT system as DT-IoT devices (the DT-IoT Layer in Fig. 5).

DRL agents operating within the DT system are not mandated to achieve optimal solutions in every iteration. Instead, they can generate a sequence of high-quality solutions through DT simulation. Here, *DeepRB* allows the DT-IoT devices to interact with each other at most Z iterations in the DT simulation at beginning of each time slot. Therefore, our focus lies on identifying the action that minimizes energy consumption by the end of the simulation. In *DeepRB*, the best action for time slot t is selected as follows:

$$\mathbf{a}^*(t) = argmax_{z\in[1,Z]} \sum_{n=1}^{N} \mathrm{R}(s_n(t), a_n(z)) \tag{20}$$

D. Solution: Clustering and Knowledge-Aware Aggregation

To address above issues, *DeepRB* proposes a novel distributed training algorithm, which empowers each IoT device to autonomously and collaboratively manage its learning agent with clustering and knowledge-aware model aggregation.

1) Clustering: Considering that IoT devices can vary greatly in their attributes, using a single global model for model aggregation may face low training efficiency. In *DeepRB*, IoT devices are grouped into distinct clusters based on their locations. Each cluster holds a shared cluster-specific model, which is then updated based on the aggregated local updates from IoT devices

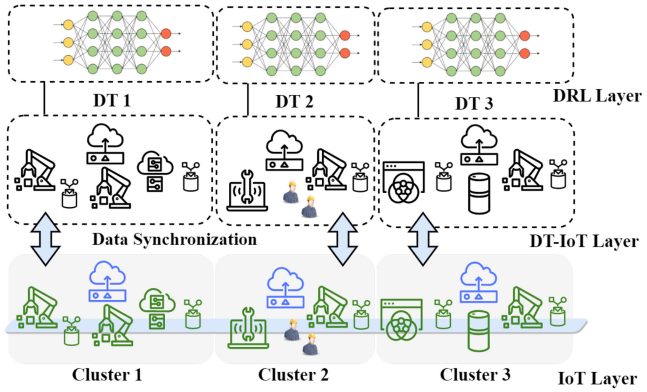


Fig. 5. DT-empowered multi-agent federated training with clustering.

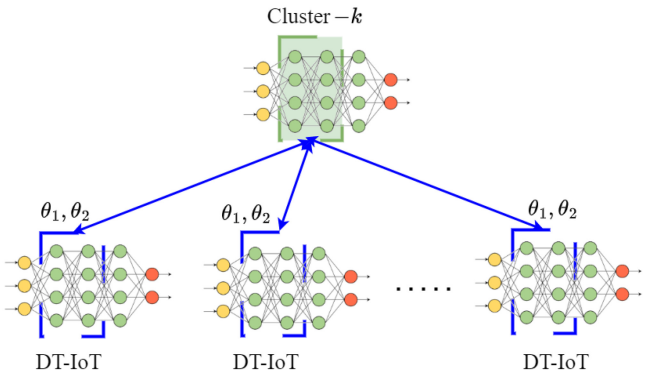


Fig. 6. Knowledge-aware model aggregation which only fuses the first two layers in DRL models. θ_1 and θ_2 are the weights of the first and second layer.

within this cluster. Let us define $\boldsymbol{\psi}_n = [\psi_{n,1}, \ldots, \psi_{n,K}]$ as the distance vector of IoT device n. We then apply the *K-means* clustering algorithm based on $[\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_N]$ to divide IoT devices into K clusters. We use variable $\eta_n \in \mathcal{K}$ to denote the cluster membership of IoT device n. It is assumed that IoT devices belong to the same cluster k, their twin objects (include DRL agents) will also be maintained in the k-th HAP (as shown in Fig. 5). With IoT devices of the same cluster having their twins hosted on the same HAP, this clustering can effectively eliminate the transmission cost of model parameters during the aggregation process.

2) Knowledge-Aware Model Aggregation: In standard model aggregation for FL, all parameters in local models are included in the average calculations. However, this approach can be counterproductive for DRL, given the heterogeneous attributes of DRL agents and their unbalanced experience replay. *DeepRB* adopts the concept of transfer learning (TL). In TL, the feature extractor (early layers) is often the main component that is transferred from a pre-trained model to a new domain. This is because the early layers usually capture general features, which are common across many domains. However, the ending layers are often replaced or fine-tuned to adapt to the new domain. In the context of *DeepRB*, the analogy extends to likening the domain concept to an IoT device. Therefore, as shown in Fig. 6, *DeepRB* only fuses the first two layers (give that most of DRL models have three layers) during the cluster-specific model aggregation.

E. Algorithm Design: DeepRB

In Algorithm 2, *DeepRB* initials a set of cluster-specific DRL models denoted by $\{\boldsymbol{\varphi}^1_{cluster}, \ldots, \boldsymbol{\varphi}^K_{cluster}\}$, where IoT devices are grouped into K clusters based on their distances to HAPs. The IoT devices in the cluster k, use $\boldsymbol{\varphi}^k_{cluster}$ to initialize their local DRL agents (denoted by $\boldsymbol{\varphi}^n_{local}$) hosted by the corresponding DT-IoT devices in DT system. For each time slot t, *DeepRB* contains four procedures, namely $DT\ Simulation$, *Actuator*, $Local\ Update$ and $Model\ Aggregation$.

In $DT\ Simulation$ (line 10 to line 16), *DeepRB* allows DT-IoT devices to interact with DT environment for Z iterations, and select an optimal action $\mathbf{a}^*(t)$ based on Eq. (20). Then, in *Actuator* (line 17 to line 19), the DT-IoT devices will send $\mathbf{a}^*(t)$ to their physical counterparts to establish real connections with HAPs. The HAPs in the same time perform resource allocation using the proposed *ResMLP* model (Section IV-B). On the other hand, the $Local\ Update$ procedure (line 20 to line 23) will be executed every t^L time slots, where each DT-IoT device $n \in \mathcal{N}$ uses its own experience replay $\mathcal{D}_n$ to train local model $\boldsymbol{\varphi}^n_{local}$ and clear its $\mathcal{D}_n$. Finally, the *DeepRB* performs $Model\ Aggregation$ (line 24 to line 32) every t^G time slots ($\geq t^L$). In this procedure, only the first two layers (denoted by $\boldsymbol{\varphi}^n_{local}[\theta_1]$ and $\boldsymbol{\varphi}^n_{local}[\theta_2]$) are aggregated.

F. Complexity Analysis

The computational complexity of *DeepRB* depends directly on the number of active IoT devices and the number of HAPs. For resource allocation, the *IRA* algorithm has a time complexity of

$$O\left(\frac{K}{\epsilon^2} \times (3N + N\log_2(\frac{1}{\xi}))\right) \tag{21}$$

where ϵ is the tolerance for subgradient method and ξ is the tolerance for bisection method which may be needed to solve $f_{n,k}(t)$ and $w_{n,k}(t)$ [36]. However, since we use the *ResMLP* model instead of the *IRA* algorithm, the time complexity of resource allocation through model inference is only $O(K)$. According to [37], the time complexity of DRL algorithms can be represented as $O(training\ steps)$. Therefore, the overall complexity of *DeepRB* is $O(TZKN)$, where T indicates the number of training episodes and Z represents the maximum number of training steps per episode.

In fact, we should focus on the training efficiency of the proposed models. The training efficiency and robust convergence of both the *ResMLP* and DRL models are evaluated and verified in the following sections.

Algorithm 2: *DeepRB*

1 # Initialization: input a set of HAPs $\mathcal{K}$ and a set of IoT devices $\mathcal{N}$.
2 ■ $Clustering$:
3 Compute distance vector $\boldsymbol{\psi}_n = [\psi_{n,1}, ..., \psi_{n,K}]$ for every $n \in \mathcal{N}$.
4 Use *K-means* to divide IoT devices into K clusters and configure cluster membership $[\eta_1, ..., \eta_N]$.
5 Create K cluster-specific DRL models $\boldsymbol{\varphi}^k_{cluster}, \forall k \in \mathcal{K}$.
6 Create N local DRL models $\boldsymbol{\varphi}^n_{local} = \boldsymbol{\varphi}^{\eta_n}_{cluster}, \forall n \in \mathcal{N}$.
7 Set DT-IoT device's experience replay $\mathcal{D}_n = \emptyset, \forall n \in \mathcal{N}$.
8 **for** ***each*** *time slot* t:$1 \rightarrow +\infty$ **do**
9 Observe IoT device state $s_n(t), \forall n \in \mathcal{N}$.
10 ■ $DT\ Simulation$:
11 **for** *each iteration* $z \in \{1, 2, ..., Z\}$ **do**
12 Each DT-IoT device n chooses an action $a_n(z) \in \mathcal{A}$ using its policy network $\boldsymbol{\varphi}^n_{local}$.
13 According to $\mathbf{a}_k(z)$, each HAP k runs *ResMLP* to solve $\mathcal{P}_k(z)$ and obtains $\mathbf{f}_k(z)$ and $\mathbf{w}_k(z)$.
14 All DT-IoT devices adjust their transmission power based on Eq. (12).
15 All DT-IoT devices obtain a reward based on Eq. (19) and store transition $(s_n(z), a_n(z), \mathrm{R}(s_n(z), a_n(z)), s_n(z+1))$ in their local experience replay $\mathcal{D}_n,\ \forall n \in \mathcal{N}$.
16 Obtain optimal action $\mathbf{a}^*(t)$ by Eq. (20).
17 ■ $Actuator$:
18 Each IoT device $n \in \mathcal{N}$ connects to the HAP recommend by its twin DT-IoT based on $\mathbf{a}^*(t)$.
19 HAPs $k \in \mathcal{K}$ perform resource allocation using proposed *ResMPL* model (Section IV-B).
20 ■ $Local\ Update$:
21 **if** $t\ \%\ t^L == 0$ **then**
22 All local models $\boldsymbol{\varphi}^n_{local}, \forall n \in \mathcal{N}$ are trained with their experience replay $\mathcal{D}_n$.
23 Set $\mathcal{D}_n = \emptyset,\ \forall n \in \mathcal{N}$.
24 ■ $Model\ Aggregation$:
25 **if** $t\ \%\ t^G == 0$ **then**
26 **for** *each cluster* $k \in \{1, 2, ..., K\}$ **do**
27 Set $\mathcal{N}^k_{cluster} = \{n | \eta_n = k, \forall n \in \mathcal{N}\}$
28 $\boldsymbol{\varphi}^k_{cluster}[\theta_1] = \frac{1}{|\mathcal{N}^k_{cluster}|} \sum_{n \in \mathcal{N}^k_{cluster}} \boldsymbol{\varphi}^n_{local}[\theta_1]$
29 $\boldsymbol{\varphi}^k_{cluster}[\theta_2] = \frac{1}{|\mathcal{N}^k_{cluster}|} \sum_{n \in \mathcal{N}^k_{cluster}} \boldsymbol{\varphi}^n_{local}[\theta_2]$
30 **for** *each DT-IoT device* $n \in \{1, 2, ..., N\}$ **do**
31 $\boldsymbol{\varphi}^n_{local}[\theta_1] = \boldsymbol{\varphi}^{\eta_n}_{cluster}[\theta_1]$
32 $\boldsymbol{\varphi}^n_{local}[\theta_2] = \boldsymbol{\varphi}^{\eta_n}_{cluster}[\theta_2]$

VI. Performance Evaluation

Next, we present emulation results to validate the schedulability and scalability of *DeepRB* when handling a large number of IoT devices. *DeepRB* is implemented using Python and the PyTorch framework. The experiments in this section are designed to achieve the following objectives: 1) evaluate the training and inference efficiency of the *ResMLP* model, 2) determine the optimal learning rate and model size for the DRL model, and 3) compare the performance of *DeepRB* against competing algorithms.

A. Parameter Settings

We consider a service area of $1000 \times 1000\ m^2$, where 8 HAPs are located in fixed locations and 52 to 63 IoT devices are randomly distributed. The maximum number of IoT devices served by each HAP is $N^{max} = 10$. The frame rate of IoT applications follows $\beta_n \sim U(5, 10)\ fps$. As discussed in Section III-B2, we adopt a log-distance path-loss model, which follows [38]. The path-loss from IoT device n to the HAP k is estimated by $128.1 + 10\sigma \log_{10}(\frac{\psi_{n,k}}{\psi_0})$. We set $\sigma = 3.75$ in response to the urban area. $\psi_{n,k}$ is the distance between IoT device and the selected HAP and ψ_0 is the reference distance (1 km). The channel background noise is set to $N_0 = -174\ dbm/hz$. The maximum transmission power of IoT devices is $P^{max} = 1\ watts$. The state transition probability of IoT devices is set to $\rho = 0.1$ and $\rho = 0.2$ (from active to idle). The channel bandwidth of HAPs is uniformly distributed, with $W_k \sim U(5, 15)\ Mhz$ for all

TABLE III
PPO NETWORK PARAMETERS

Name	Value
Number of Layers in Actor and Critic Network	3
Number of Parameters in Each Layer	64
Learning Rate	0.0001
Local Model Update Interval t^L	2
Model Aggregation Interval t^G	4
Update Epochs / Clip Parameter	10 / 0.2
Discount Factor γ	0.9
Number of Iterations in DT Simulation Z	256

$k \in \mathcal{K}$. The computational capacity of HAPs, denoted as Y_k, is normalized to range between 1 and 2 times the capacity of the *Nvidia® GeForce RTX 2060*, which has 1,920 CUDA cores. In Algorithm 1, we set $j_{max} = 2000$ and $\epsilon = 10^{-3}$. The configurations of the PPO network are summarized in Table III. The update epochs indicate the number of epochs per training update, while the clip parameter represents the epsilon value used in the clipped objective function of the PPO algorithm.

B. ResMLP-Based Resource Allocation

The successful implementation of *DeepRB* relies on the assumption that the system has prior knowledge of the edge computing architecture, including both its hardware and software components. This assumption is practical, as the architecture is specifically designed for a defined set of IoT devices and video analytics tasks, rather than being open to general use. In this context, collecting application-specific profiling data ensures the effective utilization of *ResMLP* for resource allocation.

1) Application-Specific Profiling: To create the mathematical models that is required to construct twin objects, i.e., IoT devices and their applications in the DT simulations, we conduct real-world experiments using *YOLOv5* [39] on an edge computing testbed. Specifically, we conduct extensive experiments by selecting various combinations of model versions, frame resolutions (where u is chosen from the range of 128 to 648), number of physical resource blocks (PRBs), and GPU utilization (representing the realization of system variables). The detailed descriptions of these experiments are provided in our previous work [40].

$$\begin{cases} C_{yolov5x}(u) = 1.06\text{e}{-10}u^3 + 0.017 \\ C_{yolov5l}(u) = 6.49\text{e}{-11}u^3 + 0.015 \\ C_{yolov5m}(u) = 5.51\text{e}{-11}u^3 + 0.012 \\ C_{yolov5s}(u) = 4.16\text{e}{-11}u^3 + 0.010 \\ D(u) = 16u^2 \end{cases} \tag{22}$$

Subsequently, the above mathematical models of computational complexity and data size are formulated based on the obtained experimental results.

2) Data Preparation: Based on the obtained analytic models, we artificially synthesize a large amount of data to train the proposed *ResMLP* model. Each training data point is constructed as follows: first, a test scenario (i.e., a service area consisting of 8 HAPs and 52 IoT devices) is chosen. Next, one HAP is randomly selected from the chosen scenario. Then, a random subset of IoT devices is selected to be connected to the chosen HAP, with the subset size limited to $N^{max} = 10$. Finally, ten different subsets are replicated by arbitrarily switching the order of IoT devices in the subset. These steps are repeated until the size of training dataset reaches 10^6, where the ground truths are produced by running the *IRA* algorithm, i.e., Algorithm 1. Note that due to the existence of permutations and combinations, the total number of possible data points is significantly larger than our training dataset.

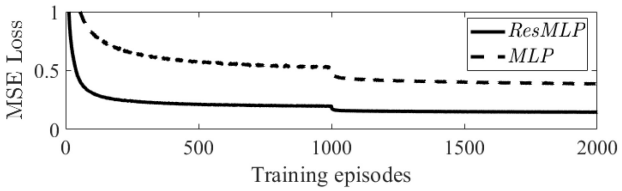


Fig. 7. The *MSELoss* of *ResMLP* and standard *MLP* model.

3) ResMLP vs Standard MLP: We compare the training efficacy of *ResMLP* with a standard *MLP*. Both models are trained with a batch size of 128 using the *Adam* optimizer, with an initial learning rate of 10^{-4} and a weight decay of 10^{-4}. The *MSELoss* function is used as the loss criterion. The learning rate is decreased by a factor of 10 every 1000 training episodes using the *MultiStepLR* strategy. As shown in Fig. 7, the loss of *ResMLP* is significantly reduced when compared with standard *MLP* model.

4) Validation of ResMLP in Resource Allocation: To evaluate the performance of the trained *ResMLP* model for fast and energy-efficient resource allocation, we compare its outputs with those of the *IRA* algorithm. Since both algorithms are used for the resource allocation of HAPs, this part of the experiments is conducted on a GPU-enabled Dell workstation, representing a typical edge server in terms of computing power.

The results are presented in Fig. 8. We first demonstrate the high efficiency of *IRA* compared to the *FAIR* algorithm, a naive resource allocation approach that evenly assigns HAP resources to the connected IoT devices. In Fig. 8 (a), we can observe notable advantages of *IRA* in improving the task completion ratio. Compared to *FAIR* algorithm, *IRA* gains an enhancement of approximately 25.8%. After validating the *IRA* algorithm, we proceed to evaluate the proposed *ResMLP* model in three aspects. Firstly, the prediction errors in Fig. 8 (b) are within 3.9% to 7.1%, which captures the average discrepancy between the output values of the *ResMLP* model and the *IRA* algorithm (the outcome of computational and network resource allocation, i.e., $\mathbf{f}_k(t) = [f_{1,k}(t), \ldots, f_{N^{max},k}(t)]$ and $\mathbf{w}_k(t) = [w_{1,k}(t), \ldots, w_{N^{max},k}(t)]$). Secondly, as depicted in Fig. 8 (c), the energy consumption of the *ResMLP* model is only increased by a maximum of 3.16% compared to the *IRA* algorithm (caused by prediction errors). Thirdly, the *ResMLP* model demonstrates remarkable acceleration in decision-making relative to the *IRA* algorithm. As illustrated

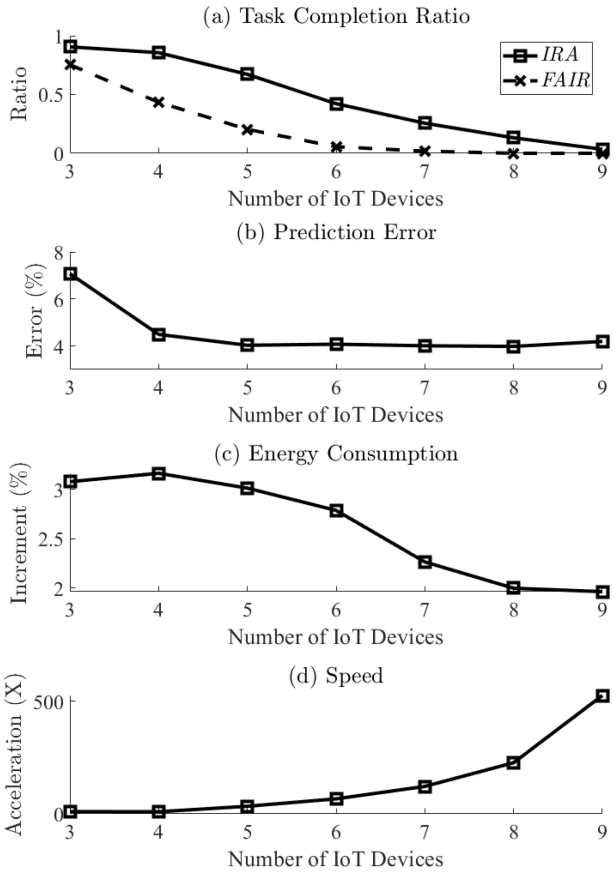


Fig. 8. Performance of *ResMLP* model against *IRA* algorithm Algorithm 1, experiments are performed on a Dell workstation with *13th Gen Intel® Core™ i9-13900K* and *Nvidia® RTX A6000*. The acceleration will be different if it is running on other machines.

in Fig. 8 (d), it exhibits a notable improvement of 10 to 500 times in acceleration (utilizing *Nvidia® RTX A6000*), providing crucial support for training our DRL agent in the subsequent step.

C. Baseline Algorithms

We perform and compare *DeepRB* with the following algorithms.

1) *Effect*: Game-based service placement algorithms, such as [41], [42]. In *Effect*, IoT devices are initially connecting to the nearest HAP. Then, the unilateral update and best response strategies are applied. Specifically, in each iteration, we randomly select one IoT device and let it switch its current selected HAP. *Effect* terminates when it reaches a Nash Equilibrium (NE) or maximum iterations.
2) *FL-1*: standard federated learning, the model aggregation process involves fusing all layers from all DRL agents into a unified model. Similar works can be found in [43], [44].
3) *FL-k* ($k > 1$): IoT devices are clustered into k groups, and during the model aggregation process, all layers from DRL agents within each cluster are fused into a cluster-specific model.

TABLE IV
COMPETING ALGORITHMS

	Model Aggregation	Clustering	*ResMLP*	PPO
Effect			√	
FL-1	√		√	√
FL-k	√	√	√	√
None-FL			√	√
DeepRB	√	√	√	√

TABLE V
IMPACT OF LAYER WIDTH AND LEARNING RATE

Parms.	$lr = 10^{-2}$	$lr = 10^{-3}$	$lr = 10^{-4}$	$lr = 10^{-5}$
32	0.0177	0.0167	0.0352	0.4287
64	**0.0159**	**0.0149***	**0.0145***	0.4169
128	0.0413	0.0192	0.0174	0.3525
256	0.1786	0.0199	0.0185	0.1901
512	0.5093	0.0191	0.0170	0.0215

4) *None-FL*: Decentralized learning, such as [45], [46]. In this approach, IoT devices autonomously train their DRL agents individually, without the need for any model aggregation.
5) *DeepRB*: ours.

All algorithms, except *Effect*, are powered by DRL (specifically, the PPO algorithm). The detailed functioning comparisons of the competing algorithms are summarized in Table IV.

D. Configuring Learning Rate and Layer Width of DRL Agent

In this subsection, we analyze how the learning rate and layer width (number of parameters in each fully connected layer) affect the training efficiency of *DeepRB*'s DRL agents with fixed number of IoT devices. The results, summarized in Table V (learning rate is denoted by *lr*), represent the average energy consumption of 52 IoT devices. From the given table, we can make several observations about the impact of these parameters on energy saving performance. Other configurations, such as local update interval t^L and model aggregation interval t^G, may also impact the performance of *DeepRB*. However, these aspects are beyond the scope of this article. For further reference, readers can consult works such as *EdgeFed* [47] and *FedSA* [48].

In Table V, it is observed that there is no clear monotonic relationship between learning rate, layer width, and energy consumption. In each row of Table V, a high learning rate can accelerate learning but may lead to overestimation and increased energy consumption. Conversely, a low learning rate can enhance stability but slow down learning, also resulting in high energy consumption. On the other hand, as illustrated in each column of Table V, layer width affects the model's capacity. Narrower layers may underfit, resulting in slower learning and higher energy consumption, while wider layers

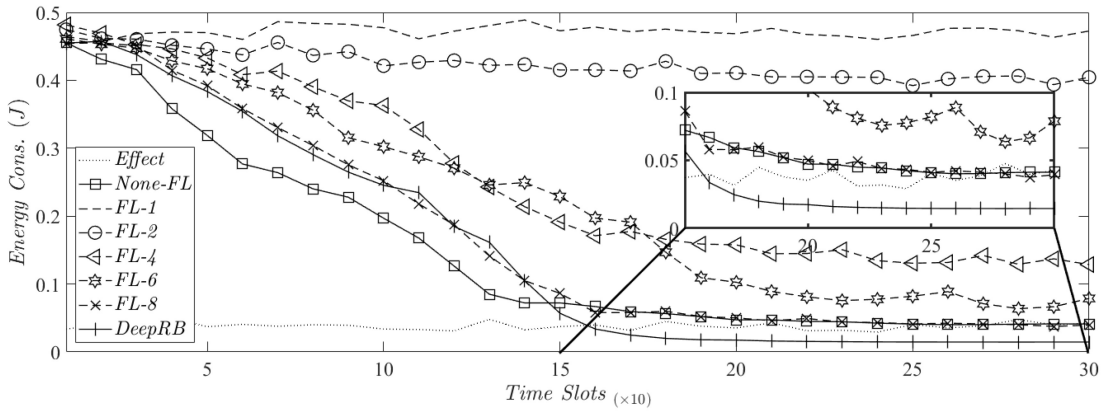


Fig. 9. Performance of *DeepRB* against competing algorithms with fixed number of IoT devices ($N = 52$).

may overfit, especially at higher learning rates, also increasing energy consumption. Overall, narrower layers generally exhibit lower energy consumption compared to wider layers at higher learning rates. However, at a very small learning rate, such as 10^{-5}, the energy consumption is lowest for the largest layer width of 512.

In Table V, the best performance of *DeepRB* is achieved with 64 parameters across different learning rates. Specifically, the combination of 64 parameters and a learning rate of 10^{-4} or 10^{-3} yields the lowest energy consumption of $0.0145J$ (or $0.0149J$), indicating optimal performance (by the end of training).

E. Short-Term Performance Comparisons

This subsection evaluate the performance of *DeepRB* in terms of convergence and energy saving against competing algorithms. The results are depicted in Fig. 9, where *DeepRB* and *FL-8* (with 8 clusters), and *None-FL* can quickly converge within 200 time slots.

Since the experience replay of IoT devices can be highly unbalanced, fusing all the local models into one global model can be counterproductive. Standard federated learning (*FL-1*) with a single global model often fails to learn effectively. However, as the number of clusters increases (from *FL-2* to *FL-8*), model aggregation becomes more effective. When the number of clusters equals 8 (the number of HAPs), the *FL-8* algorithm achieves performance similar to that of *None-FL*, where IoT devices train their DRL agents individually without any model aggregation. In Fig. 9, both *FL-8* and *None-FL* algorithms result in an average energy consumption of $0.040J$ (by the end of training). In contrast, the game-based *Effect* algorithm, which utilizes unilateral updates and best response strategies, achieves a more competitive performance with an energy consumption of $0.038J$. However, *Effect* is not stable, as the random selection of IoT devices during its update process can lead to unpredictable outcomes. In this evaluation, the *Effect* algorithm experiences a standard deviation of $0.012J$, whereas *FL-8* and *None-FL* maintain nearly zero standard deviation over time.

Our approach, *DeepRB* not only groups IoT devices into 8 clusters based on their distances to HAPs, but also employs knowledge-aware model aggregation by fusing only the feature extractors of the actor and critic models (the first two layers). Thus, compared to *None-FL*, *DeepRB* allows for knowledge sharing; while compared to *FL-8*, *DeepRB* enables IoT devices to preserve their own independent characteristics (in their DRL agents). In Fig. 9, *DeepRB* gains remarkable performance improvements. Compared to *FL-8* and *None-FL*, it reduces energy consumption by approximately 64%. In comparison with *Effect*, *DeepRB* achieves a saving of 62%.

F. Long-Term Performance Comparisons

In this subsection, we evaluate the long-term performance of *DeepRB*, focusing on scenarios where the number of active IoT devices dynamically changes with probability ρ. Specifically, each IoT device may remain in idle state with a probability of ρ for various reasons. These reasons include insufficient data collection in current time slot or system-driven decisions to prioritize resources by temporarily shutting down certain IoT devices during periods of high resource competition.

1) Energy Saving: The energy consumption results of long-term performance with $\rho = 0.1$ are depicted in Fig. 10, we can easily observe that all standard federated learning algorithms generally yield poor performance under these conditions. Compared to *FL-8* and *Effect*, *DeepRB* reduces energy consumption by 43.3% and 37.2%, respectively. When compared to *None-FL*, the best-performing algorithm among the competing algorithms, *DeepRB* achieves an approximate 25.1% improvement in energy savings. As shown in Fig. 11, we also present a scenario with relatively high dynamism by setting ρ to 0.2 and increasing the total number of IoT deivces to 63. Compared to Fig. 10, Fig. 11 exhibits more significant fluctuations. With $\rho = 0.2$, *DeepRB* continues to outperform all competing algorithms. The enhancements are 44.6%, 29.8% and 22%, when compared to *FL-8*, *Effect* and *None-FL*, respectively.

2) Schedulability: In addition to energy saving, we also focus on task schedulability, which is the probability that a task can be processed within its deadline. The results are summarized in Fig. 13. It is evident that as the system dynamics increase, the schedulability of the algorithm decreases. Due to the uncertainty in its random update order, the schedulability

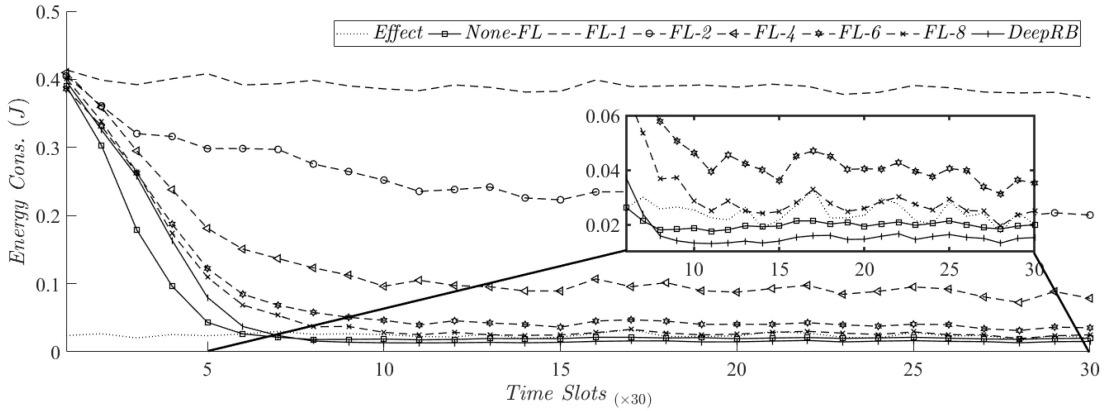


Fig. 10. Performance of *DeepRB* against competing algorithms with dynamic number of IoT devices ($\rho = 0.1$).

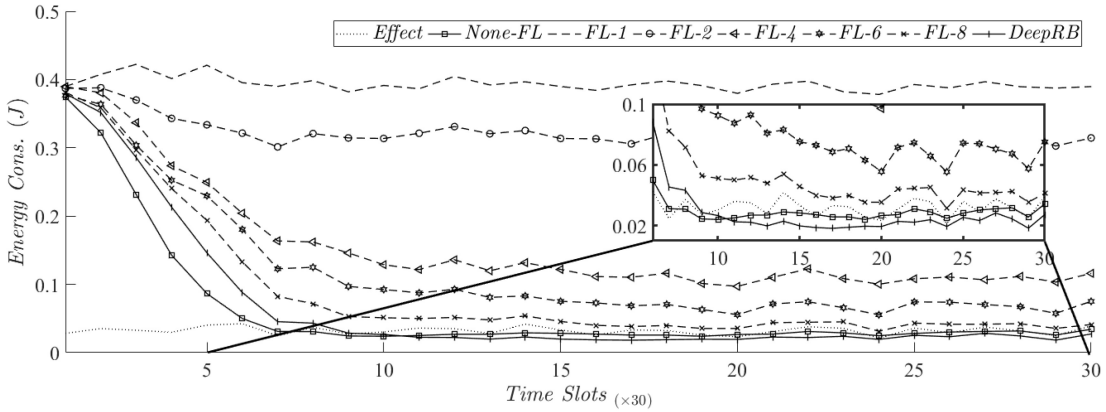


Fig. 11. Performance of *DeepRB* against competing algorithms with dynamic number of IoT devices ($\rho = 0.2$).

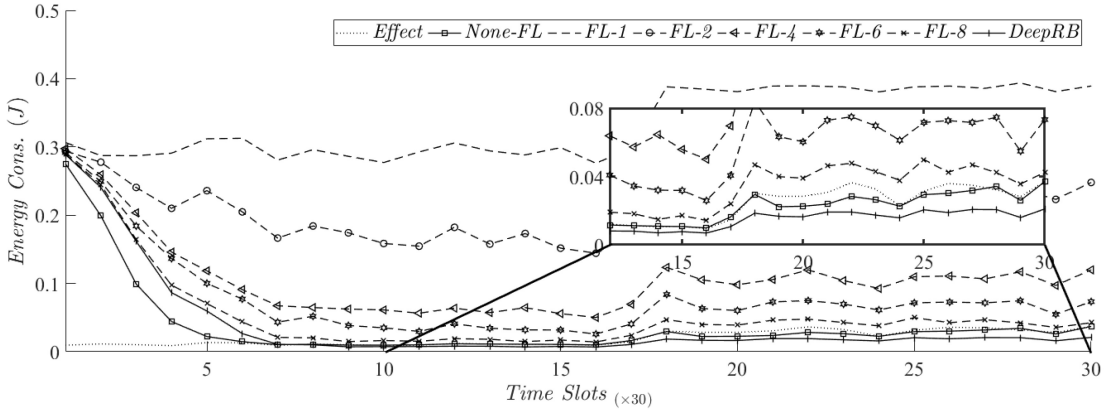


Fig. 12. Performance of *DeepRB* against competing algorithms with dynamic number of IoT devices ($\rho = 0.2$). In addition, the total number of IoT devices is increased from 56 to 63 after 500 time slots.

of the *Effect* algorithm falls between 83.5% and 85.2%, for $\rho = 0.1$ and $\rho = 0.2$, respectively. Although *FL-6* and *FL-8* demonstrate good schedulability with $\rho = 0.1$, their schedulability drops significantly when ρ increases to 0.2, indicating their poor capability in dealing with higher dynamism. In contrast, *DeepRB* and *None-FL* maintain high and stable schedulability. When $\rho = 0.2$, both algorithms can still achieve a schedulability of 0.97 and 0.95, respectively.

3) Scalability: Scalability is a critical factor in the performance evaluation of edge computing systems. In this context, scalability refers to the ability of the system to efficiently handle an increasing amount of IoT devices and adapt to changing conditions. We set $\rho = 0.2$ and initialize the system with 56 IoT devices. After 500 time slots, we increase the number of IoT devices to 63 by randomly adding 7 new devices into the service area. This setup allows us to observe how the competing algorithms and *DeepRB* adapt to this change. The results are shown in Fig. 12, where standard federated learning algorithms (*FL-k*) still exhibit poor performance. Starting from the significant fluctuation point

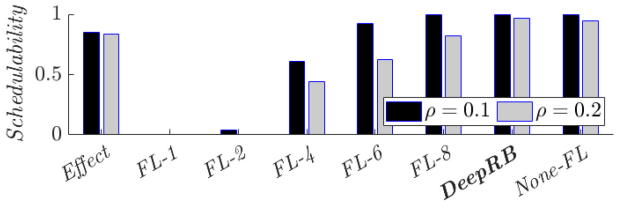


Fig. 13. Schedulability of *DeepRB* against competing algorithms with $\rho = 0.1$ and $\rho = 0.2$.

(i.e., 500 time slots), *DeepRB* reduces energy consumption by 55.1% and 40.0% when compared to *FL-8* and *Effect*, respectively. Compared to *None-FL*, *DeepRB* achieves an approximate 34.2% improvement in energy savings.

VII. Conclusion

In this paper, we presented *DeepRB*, a unified resource broker framework for real-time video analytics at edge. We showed that unlike existing state-of-the-art solutions, *DeepRB* can jointly optimize edge resource allocation and service placement towards satisfying the real-time latency requirements of video analytics applications. We demonstrated how the proposed *DeepRB* framework can be applied to large-scale real-world complex scenarios with massive IoT devices using decomposition algorithms and deep reinforcement learning methods. In *DeepRB*, we implemented a *ResMLP* network to accelerate the decision-making process in resource allocation and proposed a novel FL-assisted distributed training method to improve the model applicability. We performed extensive simulations to demonstrate *DeepRB*'s improvements towards schedulability and scalability over other traditional edge resource allocation methods. Overall, the ideas and results presented in this paper can be critical towards broader paradigm shift on edge resource management for real-time video processing.

References

[1] C. Bockelmann et al., "Massive machine-type communications in 5G: Physical and MAC-layer solutions," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 59–65, Sep. 2016.

[2] K. H. K. Reddy, G. Srivastava, R. S. Goswami, and D. S. Roy, "A hybrid optimized intelligent resource-constrained service scheduling for unified IoT applications in smart cities," *IEEE Trans. Netw. Service Manag.*, vol. 21, no. 2, pp. 1648–1659, Apr. 2024.

[3] G. Aceto, V. Persico, and A. Pescapé, "A survey on information and communication technologies for industry 4.0: State-of-the-art, taxonomies, perspectives, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3467–3501, 4th Quart., 2019.

[4] P. McEnroe, S. Wang, and M. Liyanage, "A survey on the convergence of edge computing and AI for UAVs: Opportunities and challenges," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 15435–15459, Sep. 2022.

[5] P. Chen, B. Lyu, Y. Liu, H. Guo, and Z. Yang, "Multi-IRS assisted wireless-powered mobile edge computing for Internet of Things," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 1, pp. 130–144, Mar. 2023.

[6] T. M. Hoang, S. Dinh-Van, B. Barn, R. Trestian, and H. X. Nguyen, "RIS-aided smart manufacturing: Information transmission and machine health monitoring," *IEEE Internet Things J.*, vol. 9, no. 22, pp. 22930–22943, Nov. 2022. [Online]. Available: https://doi.org/10.1109/JIOT.2022.3187189

[7] R. Rajavel, S. K. Ravichandran, K. Harimoorthy, P. Nagappan, and K. R. Gobichettipalayam, "IoT-based smart healthcare video surveillance system using edge computing," *J. Ambient Intell. Humaniz. Comput.*, vol. 13, pp. 3195–3207, Jun. 2022.

[8] E. S. A. da Silva, H. Pedrini, and A. L. Dos Santos, "Applying graph neural networks to support decision making on collective intelligent transportation systems," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 4, pp. 4085–4096, Dec. 2023.

[9] R. Trimananda, A. Younis, B. Wang, B. Xu, B. Demsky, and G. Xu, "Vigilia: Securing smart home edge computing," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, 2018, pp. 74–89.

[10] C. Qu et al., "DroneCOCoNet: Learning-based edge computation offloading and control networking for drone video analytics," *Future Gener. Comput. Syst.*, vol. 125, pp. 247–262, Dec. 2021.

[11] K. Lai, J. Lei, Y. Deng, L. Wen, G. Chen, and W. Liu, "Analyzing uplink grant-free sparse code multiple access system in massive IoT networks," *IEEE Internet Things J.*, vol. 9, no. 7, pp. 5561–5577, Apr. 2022.

[12] A. Alalewi, I. Dayoub, and S. Cherkaoui, "On 5G-V2X use cases and enabling technologies: A comprehensive survey," *IEEE Access*, vol. 9, pp. 107710–107737, 2021.

[13] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2018, pp. 756–764.

[14] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[15] H. Jiang, X. Dai, Z. Xiao, and A. Iyengar, "Joint task offloading and resource allocation for energy-constrained mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 4000–4015, Jul. 2023.

[16] K. Cheng, Y. Teng, W. Sun, A. Liu, and X. Wang, "Energy-efficient joint offloading and wireless resource allocation strategy in multi-MEC server systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2018, pp. 1–6.

[17] Y. Chen, B. Ai, Y. Niu, Z. Zhong, and Z. Han, "Energy efficient resource allocation and computation offloading in millimeter-wave based fog radio access networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2020, pp. 1–7.

[18] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Performance optimization in mobile-edge computing via deep reinforcement learning," in *Proc. IEEE 88th Veh. Technol. Conf. (VTC)*, 2018, pp. 1–6.

[19] N. Zhao, Y.-C. Liang, D. Niyato, Y. Pei, M. Wu, and Y. Jiang, "Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 11, pp. 5141–5152, Nov. 2019.

[20] B. Yamansavascilar, A. C. Baktir, C. Sonmez, A. Ozgovde, and C. Ersoy, "DeepEdge: A deep reinforcement learning based task orchestrator for edge computing," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 1, pp. 538–552, Jan./Feb. 2023.

[21] L. Ale, S. A. King, N. Zhang, A. R. Sattar, and J. Skandaraniyam, "D3PG: Dirichlet DDPG for task partitioning and offloading with constrained hybrid action space in mobile-edge computing," *IEEE Internet Things J.*, vol. 9, no. 19, pp. 19260–19272, Oct. 2022.

[22] P. Li, Z. Xiao, X. Wang, K. Huang, Y. Huang, and H. Gao, "EPtask: Deep reinforcement learning based energy-efficient and priority-aware task scheduling for dynamic vehicular edge computing," *IEEE Trans. Intell. Veh.*, vol. 9, no. 1, pp. 1830–1846, Jan. 2024.

[23] D. H. Abdulazeez and S. K. Askar, "Offloading mechanisms based on reinforcement learning and deep learning algorithms in the fog computing environment," *IEEE Access*, vol. 11, pp. 12555–12586, 2023.

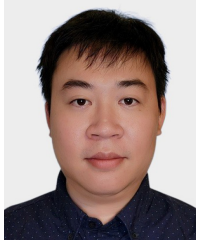
[24] D. Chemodanov, P. Calyam, F. Esposito, R. McGarvey, K. Palaniappan, and A. Pescapé, "A near optimal reliable orchestration approach for geo-distributed latency-sensitive SFCs," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2730–2745, Oct.–Dec. 2020.

[25] L. Tan, Z. Kuang, L. Zhao, and A. Liu, "Energy-efficient joint task offloading and resource allocation in ofdma-based collaborative edge computing," *IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 1960–1972, Mar. 2022.

[26] D. H. Kim, A. Manzoor, M. Alsenwi, Y. K. Tun, W. Saad, and C. S. Hong, "Ruin theory for user association and energy optimization in multi-access edge computing," 2021, *arXiv:2107.00901*.

[27] C. Zhang, H. Zhao, and S. Deng, "A density-based offloading strategy for IoT devices in edge computing systems," *IEEE Access*, vol. 6, pp. 73520–73530, 2018.

[28] X. Zhou, L. Huang, T. Ye, and W. Sun, "Computation bits maximization in UAV-assisted MEC networks with fairness constraint," *IEEE Internet Things J.*, vol. 9, no. 21, pp. 20997–21009, Nov. 2022.
[29] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2020, pp. 257–266.
[30] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for mobile edge computing," in *Proc. 16th Int. Symp. Model. Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt)*, 2018, pp. 1–6.
[31] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, Mar. 2018.
[32] B. Li, Y. Liu, L. Tan, H. Pan, and Y. Zhang, "Digital twin assisted task offloading for aerial edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 10, pp. 10863–10877, Oct. 2022.
[33] R. Dong, C. She, W. Hardjawana, Y. Li, and B. Vucetic, "Deep learning for hybrid 5G services in mobile edge computing systems: Learn from a digital twin," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4692–4707, Oct. 2019.
[34] Q. Guo, F. Tang, and N. Kato, "Federated reinforcement learning-based resource allocation for d2d-aided digital twin edge networks in 6G industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 19, no. 5, pp. 7228–7236, May 2023.
[35] T. Do-Duy, D. Van Huynh, O. A. Dobre, B. Canberk, and T. Q. Duong, "Digital twin-aided intelligent offloading with edge selection in mobile edge computing," *IEEE Wireless Commun. Lett.*, vol. 11, no. 4, pp. 806–810, Apr. 2022.
[36] X. Chen et al., "Optimal two-timescale configuration of mobile edge computing with mixed energy supply," *IEEE Trans. Smart Grid*, vol. 15, no. 5, pp. 4765–4778, Sep. 2024.
[37] H. Li, K. D. R. Assis, S. Yan, and D. Simeonidou, "DRL-based long-term resource planning for task offloading policies in multiserver edge computing networks," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 4, pp. 4151–4164, Dec. 2022.
[38] H. Gao, W. Huang, T. Liu, Y. Yin, and Y. Li, "PPO2: Location privacy-oriented task offloading to edge computing using reinforcement learning for intelligent autonomous transport systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 7, pp. 7599–7612, Jul. 2023.
[39] G. Jocher. "YOLOv5." 2020. [Online]. Available: https://github.com/ultralytics/yolov5
[40] X. Zhang, A. Pal, and S. Debroy, "EdgeURB: Edge-driven unified resource broker for real-time video analytics," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp. (NOMS)*, 2024, pp. 1–8.
[41] X. Zhang, A. Pal, and S. Debroy, "EFFECT: Energy-efficient fog computing framework for real-time video processing," in *Proc. IEEE/ACM 21st Int. Symp. Clust., Cloud Internet Comput. (CCGrid)*, 2021, pp. 493–503.
[42] C.-Y. Hsieh, Y. Ren, and J.-C. Chen, "Edge-cloud offloading: Knapsack potential game in 5G multi-access edge computing," *IEEE Trans. Wireless Commun.*, vol. 22, no. 11, pp. 7158–7171, Nov. 2023.
[43] Y. Dai, J. Zhao, J. Zhang, Y. Zhang, and T. Jiang, "Federated deep reinforcement learning for task offloading in digital twin edge networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 3, pp. 2849–2863, May/Jun. 2024.
[44] A. Ndikumana, K. K. Nguyen, and M. Cheriet, "Federated learning assisted deep Q-learning for joint task offloading and fronthaul segment routing in open ran," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 3, pp. 3261–3273, Sep. 2023.
[45] S. Hwang, H. Lee, J. Park, and I. Lee, "Decentralized computation offloading with cooperative UAVs: Multi-agent deep reinforcement learning perspective," *IEEE Wireless Commun.*, vol. 29, no. 4, pp. 24–31, Aug. 2022.
[46] Q. Cui, X. Zhao, W. Ni, Z. Hu, X. Tao, and P. Zhang, "Multi-agent deep reinforcement learning-based interdependent computing for mobile edge computing-assisted robot teams," *IEEE Trans. Veh. Technol.*, vol. 72, no. 5, pp. 6599–6610, May 2023.
[47] Y. Ye, S. Li, F. Liu, Y. Tang, and W. Hu, "EdgeFed: Optimized federated learning based on edge computing," *IEEE Access*, vol. 8, pp. 209191–209198, 2020.
[48] Q. Ma, Y. Xu, H. Xu, Z. Jiang, L. Huang, and H. Huang, "FedSA: A semi-asynchronous federated learning mechanism in heterogeneous edge computing," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3654–3672, Dec. 2021.



Xiaojie Zhang received the M.S. degree in computer science from George Washington University, USA, in 2015, and the Ph.D. degree from the Graduate Center, City University of New York, USA, in 2023. He is currently with the School of Electronics and Information Engineering, Hunan First Normal University, Changsha, China. His research interests include distributed computing, system optimization, and deep learning.

Saptarshi Debroy (Senior Member, IEEE) received the B.Tech. degree from the West Bengal University of Technology, India, in 2006, the M.Tech. degree from Jadavpur University, India, in 2008, and the Ph.D. degree in computer engineering from the University of Central Florida in 2014. He is currently an Associate Professor with the Department of Computer Science, City University of New York. His current research interests include cyber security, distributed computing, and artificial intelligence.

Peng Wang received the Ph.D. degree in computer science from Hunan University, Changsha, China, in 2019. He is currently an Associate Professor with Hunan First Normal University, Changsha. He has published many research articles in international conference and journals. His research interests include edge computing, cloud computing, computational intelligence, and artificial intelligence.

Keqin Li (Fellow, IEEE) received the B.S. degree in computer science from Tsinghua University in 1985 and the Ph.D. degree in computer science from the University of Houston in 1990. He is a SUNY Distinguished Professor with the State University of New York and a National Distinguished Professor with Hunan University, China. He has authored or co-authored more than 1110 journal articles, book chapters, and refereed conference papers. He holds over 75 patents announced or authorized by the Chinese National Intellectual Property Administration. Since 2020, he has been among the world's top few most influential scientists in parallel and distributed computing regarding single-year impact (ranked #2) and career-long impact (ranked #4) based on a composite indicator of the Scopus citation database. He is listed in Scilit Top Cited Scholars from 2023 to 2024 and is among the top 0.02% out of over 20 million scholars worldwide based on top-cited publications. He is listed in ScholarGPS Highly Ranked Scholars from 2022 to 2024 and is among the top 0.002% out of over 30 million scholars worldwide based on a composite score of three ranking metrics for research productivity, impact, and quality in the recent five years. He received the IEEE TCCLD Research Impact Award from the IEEE CS Technical Committee on Cloud Computing in 2022 and the IEEE TCSVC Research Innovation Award from the IEEE CS Technical Community on Services Computing in 2023. He won the IEEE Region 1 Technological Innovation Award (Academic) in 2023. He was a recipient of the International Science and Technology Cooperation Award from 2022 to 2023 and the 2023 Xiaoxiang Friendship Award of Hunan Province, China. He is a Member of the SUNY Distinguished Academy. He is an AAAS Fellow, an AAIA Fellow, an ACIS Fellow, and an AIIA Fellow. He is a Member of the European Academy of Sciences and Arts. He is a Member of Academia Europaea (Academician of the Academy of Europe).