Transformer-Enhanced Task Offloading With Privacy Preservation in Heterogeneous and Dynamic Edge Networks

Zhao Tong, *Senior Member, IEEE*, Shizhen Xiao, Xi Zhang, Lihui Xia, Jing Mei, and Keqin Li, *Fellow, IEEE*

***Abstract*—The exponential growth of edge devices and growing demand for low-latency, high-throughput applications have established edge computing as essential infrastructure. Edge computing enables near-data computation to alleviate device burdens, yet task offloading optimization remains challenging due to the heterogeneous, dynamic networks and privacy risks in centralized approaches. To address these intertwined challenges, we propose the federated Q-Learning via Transformer for task offloading (FQTTO) algorithm, which integrates federated learning (FL) with a Transformer-based Q-Network. FL ensures privacy-preserving distributed model updates, while the Transformer employs customized sequential state encodings of task priorities and resource availabilities to autoregressively predict Q-values, incorporating n-step returns for long-term optimization. Extensive simulations show that compared to the baseline methods, FQTTO achieves average reductions exceeding 20.4% in delay and 22.3% in energy consumption, while improving load balance by at least 13.4% and enhancing the task completion rate by up to 12.1%.**

***Index Terms*—Deep reinforcement learning, edge computing, federated learning, multi-objective optimization, transformer.**

I. Introduction

THE burgeoning deployment of Artificial Intelligence of Things (AIoT) devices, driven by fifth-generation (5G) and the Internet of Everything (IoE), has led to an exponential surge in data generation, overwhelming the computational capabilities of edge devices [1]. Mobile edge computing (MEC) addresses this challenge by enabling computation near data sources, significantly reducing delay and bandwidth consumption compared to cloud-centric approaches [2], [3]. However, optimizing task offloading in MEC systems remains complex. This complexity arises from the heterogeneity of edges, tasks and user devices , as well as the dynamic network conditions. Additionally, stringent privacy requirements pose significant challenges, as centralized data aggregation exposes sensitive user information to potential breaches in heterogeneous networks [4], [5]. These challenges are exacerbated by the increasing scale and diversity of Internet of things (IoT) applications, which demand adaptive and scalable solutions [6], [7], [8].

To overcome these challenges, we must address two fundamental problems. On the one hand, traditional centralized deep reinforcement learning (DRL) approaches are non-viable for real-world services as they require aggregating sensitive user data, which poses unacceptable privacy risks. Federated learning (FL) offers a fundamental solution, enabling collaborative model training while preserving data locality and user privacy [9]. Therefore, integrating FL is not merely an option, but a necessity for building trustworthy MEC services. On the other hand, the decision-making policy itself must navigate a high-dimensional and dynamic state space. Traditional DRL approaches based on Multilayer Perceptrons (MLPs) struggle to capture the complex sequential dependencies inherent in MEC, such as the relationship between task priorities in a queue and time-varying resource availability. While recent architectures like the Transformer [10], have shown success in robotics by modeling sequential actions, they are not natively designed for the unique state structure of MEC. These general-purpose models lack a specific mechanism to handle the prioritized and heterogeneous nature of task queues or server congestion states.

To address these, we propose federated Q-Learning via Transformer for task offloading (FQTTO) algorithm, which integrates FL with a Transformer-based dueling Q-Network. We develop a customized Transformer architecture with a state encoding mechanism tailored specifically to tokenize and model MEC states. This allows FQTTO to learn a sophisticated policy that intelligently balances the multi-objective Quality of Service (QoS) requirements, all while operating under the privacy-by-design guarantees of FL.

A. Motivation

The motivation for this research stems from the performance limitations of current approaches, resulting in higher latency and energy overhead under heterogeneous conditions [11].

Received 8 November 2025; revised 19 May 2026; accepted 24 May 2026. Date of publication 26 May 2026; date of current version 2 September 2026. This work was supported by the Program of National Natural Science Foundation of China under Grant 62372172 and Grant 62072174, in part by the Distinguished Youth Science Foundation of Hunan Province, China under Grant 2023JJ10030. Recommended for acceptance by J. Xu.*(Corresponding author: Jing Mei.)*

Zhao Tong, Shizhen Xiao, Xi Zhang, Lihui Xia, and Jing Mei are with the College of Information Science and Engineering, Hunan Normal University, Changsha 410081, China (e-mail: tongzhao@hunnu.edu.cn; mzdsxws@hunnu.edu.cn; 202470294314@hunnu.edu.cn; xialihui@hunnu.edu.cn; jingmei@hunnu.edu.cn).

Keqin Li is with the College of Information Science and Engineering, Hunan University and National Supercomputing Center, 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/TMC.2026.3697303

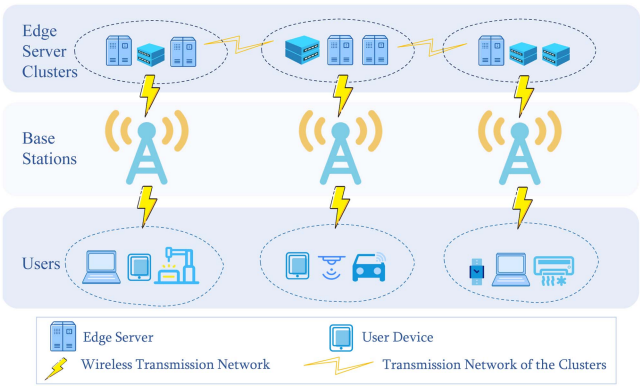


Fig. 1. Hierarchical network system integrated efficient computing for MEC.

The limitations of existing decision-making models are clear. Conventional offloading heuristics fail to adapt to dynamic networks. While DRL is promising, MLP-based methods are inadequate for capturing complex temporal dependencies in high-dimensional MEC states [12], [13]. Recent advances, such as Google's Q-Transformer, have demonstrated remarkable success in handling complex decision-making tasks in robotics environments [14]. However, these models are designed for physical action sequences, lacking a tailored architecture to understand the state representations unique to MEC task offloading. Their tailored integration to edge computing remains limited, presenting a significant research opportunity.

Furthermore, privacy is a paramount concern in service computing, as raw data exposure violates user constraints. FL mitigates this via decentralized updates, yet its integration with Transformers for task offloading is unexplored.

These gaps drive FQTTO. We fuse FL's privacy preservation with a custom-designed Transformer-encoded dueling Q-Network to optimize multi-objective offloading. The hierarchical MEC network system is shown in Fig. 1.

B. Contributions

This study makes several contributions to the field of task offloading in dynamic, heterogeneous and privacy-concerned edge computing environments, as outlined below.

- We construct a system model that accounts for mutable communication conditions, heterogeneous network architectures and task types, while integrating a dual queuing mechanism. It dynamically prioritizes tasks based on computational complexity, deadlines and resource demands, more closely mirroring the variability and constraints of real-world MEC environments.
- We introduce a load balance index to optimize the distribution of tasks among the edge servers, improving the reliability of the system and the utilization of resources. By integrating node-state information, it prevents node overload, ensuring robust task execution and improved QoS under diverse workloads.
- We propose FQTTO, a novel algorithm that integrates FL for privacy-preserving distributed training with a Transformer-based decision model. Unlike previous approaches based on MLP, this framework minimizes delay and energy consumption in dynamic task offloading MEC system, leveraging autoregressive Q-learning to handle combinatorial action spaces, outperforming conventional methods in dynamic and heterogeneous environments with privacy protection.
- We conduct extensive simulations on a dataset that inherits the characteristics of the Alibaba Cluster Trace. Through comprehensive experiments, including ablation and extension studies, FQTTO outperforms in critical indicators, including average delay and average energy consumption, conventional DRL baseline approaches. The experimentation shows the robustness and applicability of the algorithm in different scenarios.

The remainder of this paper is organized as follows. Section II analyzes the related works through different aspects. Section III presents the system model, including computation, communication, queue and load balance models. Section IV formulates the optimization problem and proves its NP-hardness. Section V details the FQTTO algorithm that integrates FL and a Transformer-based decision model. Section VI evaluates FQTTO through extensive experiments and Section VII concludes the article with future research directions.

II. Related Work

In this section, we review and analyze recent research on task offloading in MEC, categorizing it into three primary approaches: traditional optimization-based methods, DRL-based methods and Transformer-based methods.[1]

A. Traditional Optimization-Based Methods

Traditional optimization techniques, such as Lyapunov drift plus penalty theory, game-theoretic models and heuristic algorithms, provide robust frameworks for MEC with provable performance guarantees. Dong et al. [15] proposed a multi-objective particle swarm optimization (PSO)-driven optimizer, generating Pareto-optimal solutions for MEC latency-energy trade-offs. Liang et al. [21] formulated a mixed-integer nonlinear program (MINLP) for task offloading in MEC, solved via branch-and-bound with approximation guarantees. Wu et al. [16] developed a Lyapunov-based algorithm for computation offloading, optimizing energy efficiency under strict security constraints. Zhao et al. [22] extended Lyapunov methods to address MEC delay and stability, proposing a universal framework for sustainable offloading. Li et al. [17] introduced a Stackelberg game model for multi-user edge computing with FL, addressing privacy concerns.

Despite their efficacy in controlled settings, these methods often rely on convex relaxations or static assumptions, limiting adaptability to non-stationary channel dynamics or privacy-sensitive distributed execution, resulting in suboptimal performance under real-world variability. Emerging approaches, such

[1] √ denotes full support, ✗ denotes no support, *o* denotes partial support.

as DRL, mitigate these issues by enabling adaptive decision making in dynamic MEC environments.

B. DRL-Based Methods

DRL methods employ experience-driven policy learning to manage stochastic dynamics in MEC. In addition to DRL approaches, classical reinforcement learning (RL) methods without function approximation, such as tabular Q-learning and SARSA, have also been investigated in task offloading and resource allocation problems in MEC [23], [24]. These tabular methods are effective for small-scale MEC problems with discrete state and action spaces, benefiting from algorithmic simplicity and established convergence properties. However, because they rely on explicitly enumerated state-action values, their scalability and generalization are limited in dynamic heterogeneous MEC environments with varying task arrivals, wireless channels and edge resource availability. In contrast, DRL-based methods use neural function approximation to learn policies or value functions from high-dimensional observations and have been widely investigated for computation offloading in dynamic MEC systems [18], [20]. Tang et al. [18] utilized Deep Q-Network (DQN) for energy-delay optimization in edge offloading, accounting for resource allocation and time-varying channels. Zhai et al. [20] advanced this approach with Double Deep Q-Network (DDQN) in hybrid edge environments. Similarly, Ei et al. [25] applied DDQN to minimize total energy consumption in MEC systems. Zhao et al. [19] developed multi-agent DRL (MADRL) for task offloading in UAV-assisted MEC, optimizing trajectory design, task allocation and power management. Shao et al. [26] proposed a twin-delayed deep deterministic policy gradient MADRL for UAV-assisted offloading, achieving reductions in delay and energy consumption.

Although their strengths in online adaptation and partial heterogeneity modeling via state aggregation, these methods exhibit limitations, including high sample inefficiency, discrete action spaces that scale poorly with increasing task complexity and the absence of privacy mechanisms, requiring centralized raw data that violates edge locality. Additionally, their MLP-based value functions struggle to capture long-range temporal correlations or explicit device and network variations, resulting in myopic decisions in extended, highly variable scenarios.

C. Transformer-Based Methods

Transformers enhance RL via self-attention for sequential dependency modeling in offloading trajectories. Chen et al. [12] pioneered the Decision Transformer, treating RL as sequence prediction for long-horizon tasks in robotics, enabling partial credit assignment via attention. Chebotar et al. [14] developed the Q-Transformer for autoregressive Q-function approximation in offline RL, scaling to complex discrete actions but without edge-specific state encoding. Janner et al. [27] extended this with Trajectory Transformer for offline planning in continuous control.

Despite superior handling of dependencies and partial long-horizon via multi-head attention, these methods operate in centralized settings, which are unsuitable for resource-constrained edges and neglect federated privacy, exposing sensitive trajectories during training. They also failed to enforce multi-objective rewards explicitly, limiting robustness in privacy-critical, heterogeneous MEC deployments.

Recent works have also begun to explore the synergy between DRL, FL and more expressive model architectures for privacy-preserving distributed decision making. For instance, He et al. [28] proposed an attention-weighted federated DRL framework for trust management in underwater acoustic sensor networks, where the attention mechanism effectively aggregates local models while preserving data privacy. Similarly, Martin et al. [29] introduced a federated DRL approach for optimizing dual-connectivity triggering in non-stand-alone networks, training individual DRL agents locally before aggregating them into a global model to achieve network-wide performance without centralizing sensitive data.

However, most existing studies combine only two of RL, FL and sequence modeling. A unified framework that integrates privacy-preserving FL with Transformer-enhanced Q-learning for multi-objective MEC task offloading remains underexplored.

In this paper, we examine prevailing multi-objective optimization approaches for privacy-preserving task offloading in heterogeneous, dynamic edge networks with multiple users and servers. To address this problem, we propose a novel algorithm integrating FL for model training with a Transformer-enhanced Q-learning as a solution.

Table I summarizes comparisons with related works.

III. System Model

In this section, we develop a comprehensive MEC system model that rigorously captures task heterogeneity, dynamic wireless channels and resource-constrained edge servers. The model integrates five interconnected components, including system overview, computation, communication, queue and load balance models.

A. System Overview

We consider an MEC network comprising N user devices, denoted by the set $\mathcal{N} = \{1, \ldots, N\}$. The global task set is $\mathcal{T} = \bigcup_{n\in\mathcal{N}} \mathcal{T}_n$, where $\mathcal{T}_n = \{\tau_{n,1}, \tau_{n,2}, \ldots, \tau_{n,J_n}\}$ denotes the set of tasks generated by user n and $J_n = |\mathcal{T}_n|$ is the cardinality. Each task $\tau_{n,j} \in \mathcal{T}_n$ represents a computation workload such as text processing, image analysis, audio recognition or video streaming. Task arrivals at user n follow a Poisson process to reflect real-world burstiness.

The edge layer includes M servers, denoted by $\mathcal{M} = \{1, \ldots, M\}$, connected to users through a single base station using time-division multiple access (TDMA) uplink scheduling. At time slot t, let $T_m(t)$ denote the number of tasks assigned to edge server m and $T^{\text{loc}}(t)$ denote the number of tasks executed locally across all devices. Each server m operates at fixed CPU frequency f_m, memory capacity M_m and bandwidth B_m.

Fig. 2 depicts TDMA-framed interactions, including the FL process in which raw task data remains on-device and only model parameters are uploaded during periodic aggregation rounds.

TABLE I
COMPREHENSIVE COMPARISON OF TASK OFFLOADING METHODS IN MEC[1]

Article	Method	Year	Multi-Objective	Dynamic Environment	Privacy	Long-horizon	Heterogeneity
Dong *et al.* [15]	PSO	2023	*o*	✗	✗	✗	✗
Wu *et al.* [16]	Lyapunov Opt.	2024	✗	✗	✓	✗	✗
Li *et al.* [17]	Game Theory	2025	*o*	*o*	✓	✗	✓
Tang *et al.* [18]	DQN	2022	*o*	*o*	✗	✗	✗
Zhao *et al.* [19]	MADRL	2023	✓	*o*	✗	*o*	✗
Zhai *et al.* [20]	DDQN	2024	*o*	*o*	✗	✗	✗
Chen *et al.* [12]	Decision Trm	2021	✗	✓	✗	✓	*o*
Chebotar *et al.* [14]	Q-Trm	2023	✗	✓	✗	*o*	*o*
Ours	**FQTTO**	**2025**	✓	✓	✓	✓	✓

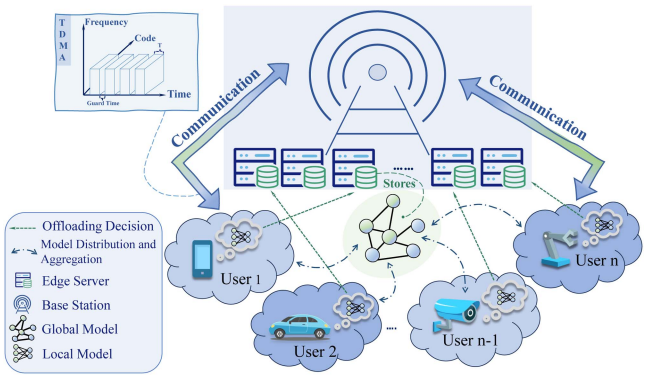


Fig. 2. Interaction framework between users and servers in edge computing environment.

B. Computation Model

Let $x_{n,j} \in \{0\} \cup \mathcal{M}$ denote the execution node for task $\tau_{n,j} \in \mathcal{T}_n$. $x_{n,j} = 0$ indicates local execution on device n, while $x_{n,j} = m$ indicates offloading to edge server m.

1) Local Execution: Local devices operate at fixed CPU frequency f_n^{loc} due to thermal and power constraints. The processing time for task $\tau_{n,j}$ is

$$T_{n,j}^{\text{loc}} = \frac{\lambda_{n,j} C_{n,j}}{f_n^{\text{loc}}} + \tau_{n,j}^{\text{w,loc}}, \tag{1}$$

where $\lambda_{n,j}$ is the input data size, $C_{n,j}$ is the computational complexity and $\tau_{n,j}^{\text{w,loc}}$ denotes the local queuing delay.

Energy consumption follows the dynamic voltage and frequency scaling model by

$$E_{n,j}^{\text{loc}} = \kappa (f_n^{\text{loc}})^2 \lambda_{n,j} C_{n,j}, \tag{2}$$

with κ as the effective switched capacitance.This formulation is consistent with the standard DVFS-based energy model adopted in MEC offloading studies [30].

2) Edge Execution: Offloaded tasks incur uplink transmission, queuing and processing delays, formulated as

$$T_{n,j}^{\text{edge}}(m) = \frac{\lambda_{n,j}}{r(m,n)} + \tau_{n,j}^{\text{w,edge}}(m) + \frac{\lambda_{n,j} C_{n,j}}{f_m}, \tag{3}$$

where $r(m,n)$ is the uplink rate, defined in Section III-C. $\tau_{n,j}^{\text{w,edge}}(m)$ is the waiting time in edge server m's queue. Downlink transmission is neglected, as result data sizes are typically small, contributing negligible delay and energy overhead.

Edge energy consumption includes processing and transmission, which can be depicted as

$$E_{n,j}^{\text{edge}}(m) = Q_m \frac{\lambda_{n,j} C_{n,j}}{f_m} + P_m \lambda_{n,j}, \tag{4}$$

where Q_m is server processing power and P_m is transmission energy per bit.

The total system energy at time slot t is

$$E_{\text{total}}(t) = \sum_{n=1}^{N} \sum_{j=1}^{J_n} \left[\mathbb{I}_{\{x_{n,j}=0\}} E_{n,j}^{\text{loc}} + \mathbb{I}_{\{x_{n,j}=m\}} E_{n,j}^{\text{edge}}(m) \right]. \tag{5}$$

C. Communication Model

Uplink transmission employs TDMA with slot duration Δt. The instantaneous uplink rate between edge server m and user n is

$$r(m,n) = B_m \log_2 \left(1 + \frac{P_n |h_{m,n}|^2}{N_0 B_m} \right), \tag{6}$$

where B_m is the allocated bandwidth, P_n is the transmit power of user n, N_0 is the noise power spectral density and $|h_{m,n}|^2 = G_0 L_{m,n} |F_{m,n}|^2$ is the channel power gain, with G_0 denoting the reference path gain. This rate model is derived from the Shannon capacity formula [31].

The path loss $L_{m,n}$ follows the urban micro model, described as

$$L_{m,n} = 10^{-(L_0 + 10\eta \log_{10} d_{m,n} + X_\sigma)/10}, \tag{7}$$

with baseline loss L_0, path loss exponent η for NLOS or LOS conditions and log-normal shadowing standard deviation X_σ.

The small-scale fading amplitude $|F_{m,n}|$ follows the Rician distribution such that

$$|F_{m,n}| \sim \begin{cases} \text{Rice}\left(\sqrt{\frac{K_{m,n}}{K_{m,n}+1}}, \sqrt{\frac{1}{2(K_{m,n}+1)}} \right), & \text{if LOS}, \\ \text{Rayleigh}\left(\sqrt{\frac{1}{2}} \right), & \text{if NLOS}, \end{cases} \tag{8}$$

with Rician factor $K_{m,n} = 3$ dB in weak LOS urban environments, ensuring $\mathbb{E}[|F_{m,n}|^2] = 1$.

D. Queue Model

The model distinguishes between local devices and edge servers. The edge queue is implemented with higher fidelity to reflect the complex dynamics of task prioritization and resource congestion, which are central to the offloading problem addressed in this work. Each server maintains a hybrid queue structure to manage task execution under resource contention.

1) Local Queue Model: Execution on the local device with $x_{n,j} = 0$ is modeled as a single-threaded system, reflecting typical resource constraints of local devices. Upon arrival, a local task $\tau_{n,j}$ enters the waiting queue $Q_w(0)$, which operates under a first-in-first-out (FIFO) discipline. The task at the head of $Q_w(0)$ is admitted to the processing queue $Q_p(0)$ only when the latter is empty. The local waiting time $\tau_{n,j}^{w,\text{loc}}$, as depicted in (1), represents the duration that $\tau_{n,j}$ spends in $Q_w(0)$ awaiting processor availability.

2) Edge Queue Model: In contrast, edge server m must manage tasks from multiple, heterogeneous users under resource contention. To handle shared resources with deadline-sensitive tasks, we implement a hybrid queue structure.

An offloaded task $\tau_{n,j}$ is enqueued in the waiting queue $Q_w(m)$ upon arrival. This queue is implemented as a priority max-heap, where task priority is defined as $p_{n,j} = 1/d_{n,j}$. This ensures that tasks with earlier deadlines and thus higher $p_{n,j}$ are prioritized for admission. The processing queue $Q_p(m)$ follows a FIFO discipline and is non-preemptive to ensure execution atomicity.

Admission from $Q_w(m)$ to $Q_p(m)$ occurs for the highest-priority task only if server m satisfies the resource and capacity constraints by

$$c_m \geq \kappa_{n,j}, \quad \mu_m \geq s_{n,j}, \quad |Q_{\mathfrak{p}}(m)| < m_m^{max}, \tag{9}$$

where c_m and μ_m denote residual CPU cycles and memory, $\kappa_{n,j}$ and $s_{n,j}$ are the task requirements and $m_m^{\max}$ is the maximum number of concurrent tasks. The edge waiting time $\tau_{n,j}^{w,\text{edge}}(m)$ is the duration spent by $\tau_{n,j}$ in $Q_w(m)$ until admission.

3) Congestion Dynamics: Congestion for edge server m at time t is updated using an exponentially weighted moving average (EWMA) by

$$\gamma_m(t) = (1-\tau)\gamma_m(t-1) + \tau \frac{|Q_p(m)|}{m_m^{\max}}, \tag{10}$$

where $\tau \in (0,1)$ is the smoothing factor. Upon congestion ($\gamma_m(t) > \theta$), transmission rates are scaled by $1/(1+\gamma_m(t))$, increasing uplink delay by a factor of $1+\gamma_m(t)$. This deterministic model captures queue-induced backpressure.

E. Load Balance Model

To achieve load balance across edge servers and improve system reliability, we define the uniformity of offloading $L(t)$ as a metric that quantifies task distribution among edge servers, enhancing resource utilization. By integrating node-state data, it prevents overload, ensuring robust execution under diverse workloads.

TABLE II
NOTATIONS

Notation	Definition
$\mathcal{N}, \mathcal{M}$	Sets of users (N) and edge servers (M)
$\mathcal{T}$	Global task set
$\mathcal{T}_n$	Task set of user n
$\tau_{n,j} \in \mathcal{T}_n$	j-th task of user n
$J_n = \lvert\mathcal{T}_n\rvert$	Number of tasks at user n
$x_{n,j} \in \{0\} \cup \mathcal{M}$	Execution node for task $\tau_{n,j}$
$\lambda_{n,j}$	Input data size of $\tau_{n,j}$ (bits)
$C_{n,j}$	CPU cycles per bit of $\tau_{n,j}$
f_n^{loc}, f_m	Local device / edge server CPU frequency (Hz)
κ	Effective switched capacitance
Q_m	edge server processing power (W)
P_m	transmission energy per bit (J/bit)
$\tau_{n,j}^{\text{w,loc}}$	Local waiting time for $\tau_{n,j}$
$\tau_{n,j}^{\text{w,edge}}(m)$	Edge waiting time for $\tau_{n,j}$ at server m
$d_{n,j}$	Deadline of task $\tau_{n,j}$
$p_{n,j} = 1/d_{n,j}$	Priority score of $\tau_{n,j}$
$Q_w(m), Q_p(m)$	Waiting / processing queue at server m
$Q_w(0), Q_p(0)$	Waiting / processing queue at local device
$m_m^{\max}$	Maximum processing capacity at server m
$\gamma_m(t) \in [0,1]$	Congestion index at server m at time t
$\tau \in (0,1)$	EWMA smoothing factor
B_m	Bandwidth of server m (Hz)
P_n	Transmit power of user n (W)
N_0	Noise power spectral density (W/Hz)
$h_{m,n}$	Complex channel coefficient
$L_{m,n}$	Path loss between m and n
$\lvert F_{m,n}\rvert$	Small-scale fading amplitude (Rice or Rayleigh)
$r(m,n)$	Uplink transmission rate (bit/s)
$T_m(t)$	Total tasks at server m at time t
$L(t) \in [0,1]$	Load-balance index at time t
ϵ	Numerical stability constant

At time slot t, let $T_m(t) = |Q_p(m)| + |Q_w(m)|$ denote the total number of tasks at edge server m, where $Q_p(m)$ and $Q_w(m)$ are the processing and waiting queues, respectively. The mean task count per edge server is

$$\mu(t) = \frac{\sum_{m\in\mathcal{M}} T_m(t)}{M}. \tag{11}$$

The variance of the task distribution is

$$\sigma^2(t) = \frac{1}{M}\sum_{m\in\mathcal{M}} (T_m(t) - \mu(t))^2. \tag{12}$$

The load balance index is

$$L(t) = \max\left(0, \min\left(1, 1 - \frac{\sqrt{\sigma^2(t)}}{\mu(t)+\epsilon}\right)\right), \tag{13}$$

where higher $L(t)$ indicates a more uniform task distribution, reducing overload risk. By optimizing $L(t)$, our method ensures balanced resource utilization and system stability in dynamic MEC environments.

A summary of key notations is given in Table II.

IV. Problem Description

In this section, we formally formulate the multi-objective task offloading optimization problem in the heterogeneous MEC

environment. The objectives are to minimize the total system delay and the energy consumption while maximizing load balance, subject to constraints such as resource limitations and task processing order. We prove the problem is NP-hard and provide a mathematical formulation amenable to RL solutions.

The decision variables are the offloading assignments $x_{n,j} \in \{0\} \cup \mathcal{M}$ for each task $\tau_{n,j} \in \mathcal{T}_n$. The total latency for task $\tau_{n,j}$ is

$$T_{n,j}^{\text{total}}(x_{n,j}) = \mathbb{I}_{\{x_{n,j}=0\}} T_{n,j}^{\text{loc}} + \mathbb{I}_{\{x_{n,j}=m\}} \left(\frac{\lambda_{n,j}}{r(m,n)/(1+\gamma_m(t))} + \tau_{n,j}^{\text{w,edge}}(m) + \frac{\lambda_{n,j} C_{n,j}}{f_m} \right). \tag{14}$$

where the uplink rate is scaled by congestion $\gamma_m(t)$ from (10) and $\tau_{n,j}^{\text{w,edge}}(m)$ depends on the max-heap priority queue order.

The system-wide average delay at time t is

$$D(t) = \frac{1}{|\mathcal{T}|} \sum_{n=1}^{N} \sum_{j=1}^{J_n} T_{n,j}^{\text{total}}(x_{n,j}), \tag{15}$$

with $|\mathcal{T}| = \sum_n J_n$. The total energy $E_{\text{total}}(t)$ is given in (5).

The multi-objective optimization problem **P1** is

$$\begin{aligned} \min_{\{x_{n,j}\}} \quad & D(t) + \alpha E_{\text{total}}(t) + \beta(1 - L(t)) \\ \text{s.t.} \quad & x_{n,j} \in \{0\} \cup \mathcal{M}, \quad \forall n \in \mathcal{N},\ j = 1, \ldots, J_n, \quad \text{(C1)} \\ & |Q_p(m)| \le m_m^{\max}, \quad \forall m \in \mathcal{M}, \quad \text{(C2)} \\ & \sum_{n,j:x_{n,j}=m} \lambda_{n,j} C_{n,j} \le f_m \Delta t, \quad \forall m \in \mathcal{M}, \quad \text{(C3)} \\ & \gamma_m(t) \le \theta, \quad \forall m \in \mathcal{M}, \quad \text{(C4)} \\ & T_{n,j}^{\text{total}}(x_{n,j}) \le d_{n,j}, \quad \forall n, j, \quad \text{(C5)} \\ & |Q_p(0)| \le 1, \quad \text{(C6)} \\ & c_m \ge \sum_{n,j:x_{n,j}=m} \kappa_{n,j}, \quad \forall m \in \mathcal{M}, \quad \text{(C7)} \\ & \mu_m \ge \sum_{n,j:x_{n,j}=m} s_{n,j}, \quad \forall m \in \mathcal{M}, \quad \text{(C8)} \\ & r(m,n) > 0, \quad \forall m \in \mathcal{M}, \forall n \in \mathcal{N}, \quad \text{(C9)} \\ & L(t) \ge L_{\min}, \quad \text{(C10)} \end{aligned} \tag{16}$$

where α, β are weighting factors, θ is the congestion threshold and $L_{\min}$ is the minimum load-balancing requirement. The constraints ensure assignment feasibility, resource availability, delay compliance and stable system operation. Constraint (C1) specifies that each task is executed either locally or at one edge server. Constraint (C2) limits the number of tasks concurrently admitted to the processing queue of server m. Constraint (C3) ensures that the total CPU demand assigned to server m does not exceed its per-slot computing budget. Constraint (C4) bounds the server congestion level by the threshold θ. Constraint (C5) requires each task to meet its latency deadline. Constraint (C6) reflects the single-threaded local execution model, i.e., at most one task can be processed locally at a time. Constraint (C7) constrains the aggregate computational-resource demand at server m. Constraint (C8) constrains the aggregate memory demand at server m. Constraint (C9) requires a positive uplink transmission rate for feasible task offloading. Constraint (C10) enforces a minimum level of load balancing across edge servers.

We show that **P1** is computationally intractable by observing that it contains the classical Knapsack problem as a special case.

Theorem 1: The optimization problem **P1** is NP-hard.

Proof: Consider a simplified special case of **P1** with a single edge server ($M = 1$). Each task $\tau_{n,j}$ has two possible execution choices: local execution ($x_{n,j} = 0$) or offloading to the edge server ($x_{n,j} = 1$). Assume identical deadlines $d_{n,j} = +\infty$, no congestion ($\gamma_1(t) = 0$) and negligible transmission and waiting delay. In addition, assume that the delay term is identical for all feasible decisions, so $D(t)$ becomes a constant and can be omitted from the optimization objective. Let $\alpha = 1$ and $\beta = 0$, so that the remaining objective is dominated by the energy consumption term.

For each task (n, j), define the energy difference between local execution and edge execution as

$$v_{n,j} = E_{n,j}^{\text{loc}} - E_{n,j}^{\text{edge}},$$

which represents the energy saving obtained by offloading the task. Let the computational demand of task (n, j) be

$$w_{n,j} = \lambda_{n,j} C_{n,j}.$$

Under these assumptions, selecting tasks for offloading reduces the total energy consumption by $\sum v_{n,j}$ while consuming edge computing resources $\sum w_{n,j}$. The edge server capacity constraint (C3) becomes

$$\sum_{n,j:x_{n,j}=1} w_{n,j} \le f_1 \Delta t.$$

Therefore, the simplified problem becomes selecting a subset of tasks for offloading so as to maximize the total energy saving under a capacity constraint. This is exactly a classical 0-1 Knapsack instance, where each task corresponds to an item with value $v_{n,j}$ and weight $w_{n,j}$ and the knapsack capacity is $W = f_1 \Delta t$.

Since the 0-1 Knapsack problem is NP-hard and the above simplified setting is a special case of **P1**, the general problem **P1** is NP-hard as well. □

This NP-hardness result motivates the following FQTTO design, which adopts a sequence-aware decision model and federated training to obtain an efficient and privacy-preserving approximation.

V. FQTTO Algorithm Design

In this section, we introduce the FQTTO algorithm, which integrates FL and a Q-learning via Transformer-based reinforcement framework in task offloading scenario (QTTO) to achieve adaptive and privacy-preserving offloading decisions

in dynamic, heterogeneous edge computing environments with privacy concern.

A. Reinforcement Learning Settings

The RL formulation consists of state, action and reward definitions, where the state is encoded as a structured sequence capturing server loads, task priorities and queue status.

State: The state at time step t, denoted as S_t, encapsulates the dynamic heterogeneity of the MEC system, including real-time server loads and the prioritized task backlog. It is structured as an ordered sequence of embedded tokens for Transformer-based processing, formulated as

$$S_t = \left[\mathbf{s}_1^{\text{server}}, \ldots, \mathbf{s}_M^{\text{server}}, \mathbf{s}_1^{\text{task}}, \ldots, \mathbf{s}_K^{\text{task}}\right] \in (\mathbb{R}^d)^{M+K}, \quad (17)$$

where the sequence concatenates two disjoint subsequences in fixed order, namely the server subsequence $S_t^{\text{server}} = [\mathbf{s}_1^{\text{server}}, \ldots, \mathbf{s}_M^{\text{server}}]$ followed by the task subsequence $S_t^{\text{task}} = [\mathbf{s}_1^{\text{task}}, \ldots, \mathbf{s}_K^{\text{task}}]$. This disjoint subsequence design explicitly preserves the bipartite structure of the MEC system, where servers and tasks correspond to two distinct entity groups. Compared with a fully flattened representation, the structured concatenation preserves the group-level organization of server and task tokens, allowing self-attention layers to learn dependencies among elements within the same group and across different groups. This follows the general capability of Transformer-style self-attention to model pairwise dependencies among sequence elements without recurrence or convolution [32]. Moreover, the fixed ordering of the server and task subsequences provides a stable positional reference, while self-attention helps capture interactions within and across the two groups under varying system states.

The **server subsequence** encodes each of the M servers. Specifically, the m-th token $\mathbf{s}_m^{\text{server}} \in \mathbb{R}^d$ ($m \in \mathcal{M}$) embeds server m via

$$\mathbf{f}_m^{\text{server}}(t) = \left[T_m(t),\ \gamma_m(t),\ f_m,\ B_m\right]^\top,$$

where $T_m(t) = |Q_p(m)| + |Q_w(m)|$ is the total task count at server m (11), $\gamma_m(t)$ is the congestion index (10), f_m is CPU frequency and B_m is bandwidth. Positional encodings distinguish server indices.

The **task subsequence** forms a fixed-length segment that encodes the K highest-priority pending tasks, sorted by descending priority $p_{n,j} = 1/d_{n,j}$. Top-K truncation bounds the Transformer input length and focuses decoding on high-priority tasks that dominate near-term scheduling quality. The value of K balances representation fidelity against the quadratic attention cost and is selected according to system scale. The k-th token embeds task $\tau_{n,j}$ as

$$\mathbf{f}_k^{\text{task}}(t) = \left[\lambda_{n,j},\ C_{n,j},\ p_{n,j},\ d_{n,j}\right]^\top,$$

with zero-padding or masking if fewer than K tasks are pending. From a representation perspective, the above state construction can be viewed as a task-oriented encoding, where only decision-relevant attributes are retained. This is conceptually related to semantic communication in the broad sense that task-oriented information is emphasized over raw system-state detail.

Alternative state representation strategies encompass a variety of architectural approaches. For instance, MLP-based global vector encoding often lacks relational inductive bias, while graph neural networks, despite explicitly modeling interactions, introduce significant structural and computational overhead. Another common approach involves autoencoder-based latent representations that compress the state into a low-dimensional space. However, such methods are typically optimized for reconstruction fidelity rather than decision-oriented value estimation, which may limit their effectiveness for the RL objective considered here [33], [34]. In contrast, the Transformer-based formulation learns representations jointly with Q-value prediction and is well suited to modeling long-range dependencies in our setting. It significantly outperforms traditional MLP-based method by supporting permutation-invariant yet order-aware reasoning.

Action: The action at time step t, denoted as A_t, determines the offloading destination for the current task, selecting among local execution or one of the M edge servers. The discrete action space is defined as

$$\mathcal{A} = \{0, 1, \ldots, M\}, \quad (18)$$

For neural network compatibility, each action a_j is encoded as a one-hot vector $\mathbf{a}_j \in \{0,1\}^{M+1}$. For example, $\mathbf{a}_j = [1, 0, \ldots, 0]^\top$ represents local execution, while $\mathbf{a}_j = [0, \ldots, 0, 1]^\top$ indicates offloading to server m.

Reward: The reward R_t is designed to optimize end-to-end delay, energy consumption and load balance while enforcing deadline constraints, in full alignment with the computation and queueing models. It is decomposed as

$$R_t = R_t^{\text{pos}} + R_t^{\text{neg}}, \quad (19)$$

where the positive component is a weighted sum, given as

$$R_t^{\text{pos}} = \alpha R_{\text{delay}} + \beta R_{\text{energy}} + \gamma R_{\text{balance}}, \quad (20)$$

with $\alpha, \beta, \gamma \geq 0$, $\alpha + \beta + \gamma = 1$, tuned in Section VI-A. The terms are

- $R_{\text{delay}} = 1 - \min(1, \frac{D_{n,j}(x_{n,j})}{d_{n,j}})$: normalized delay satisfaction, where $D_{n,j}(x_{n,j})$ is the total delay for task $\tau_{n,j}$ under decision $x_{n,j}$ and $d_{n,j}$ is its deadline.
- $R_{\text{energy}} = 1 - \tilde{E}_{n,j}(x_{n,j})$: normalized energy efficiency, with $\tilde{E}_{n,j}(x_{n,j}) \in [0,1]$ being the scaled energy cost (local or offloading).
- $R_{\text{balance}} = L(t)$: load balance index defined in Section III-E, based on $T_m(t)$ across $\mathcal{M}$.

The penalty term strictly enforces deadlines by

$$R_t^{\text{neg}} = \begin{cases} -\left(\frac{D_{n,j}(x_{n,j})}{d_{n,j}} - 1\right) & \text{if } D_{n,j}(x_{n,j}) > d_{n,j}, \\ 0 & \text{otherwise.} \end{cases} \quad (21)$$

By learning from this reward structure, the agent iteratively refines its policy to optimize the trade-offs between delay, energy consumption and load balance, thereby enhancing the overall performance and stability of the MEC system.

B. QTTO Decision Process

QTTO reformulates prioritized task offloading as an autoregressive sequence generation problem, adapting the Transformer

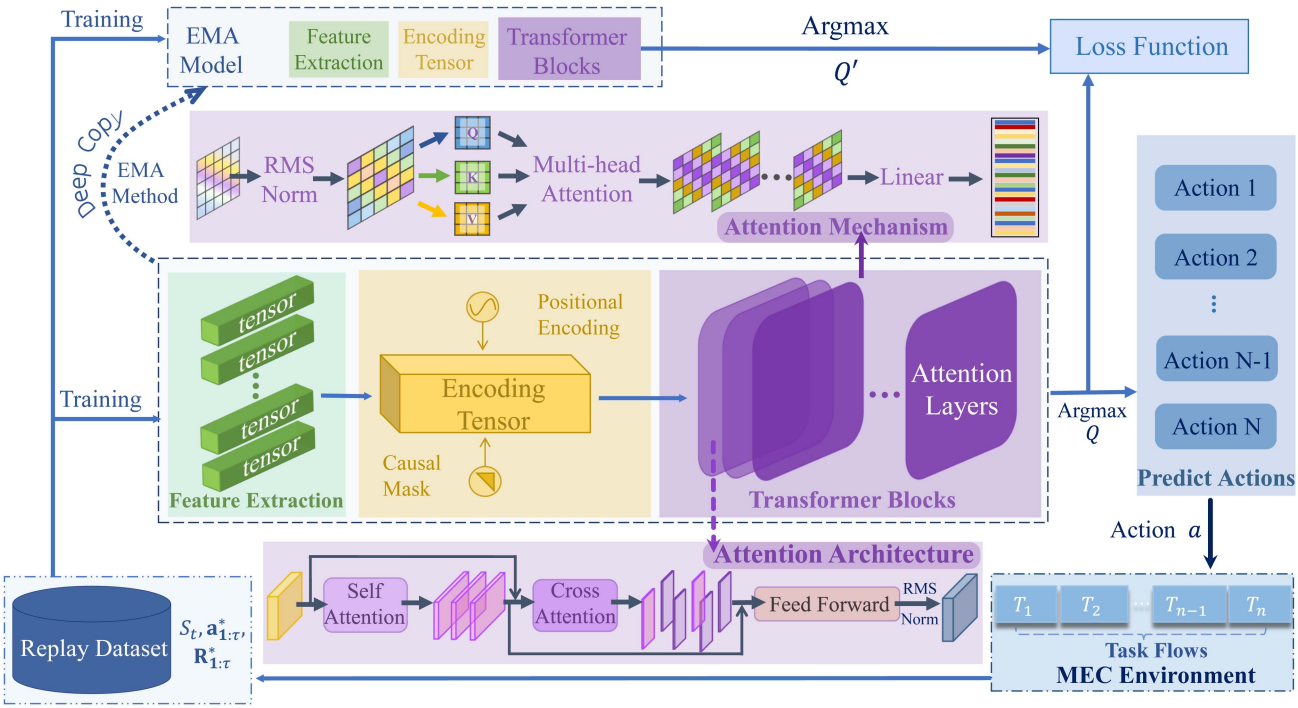


Fig. 3. Architecture of the QTTO decision process with its procedural stages and core components.

architecture via Q-learning to the dynamic and heterogeneous nature of MEC environments. As illustrated in Fig. 3, QTTO follows a progressive workflow from state tokenization of MEC observations (green and yellow), to Transformer-based sequence modeling with causal decoding (purple and pink) and finally to RL optimization through replay-based training and target updates (blue). Unlike conventional DQN methods that evaluate each task independently, QTTO employs a causal Transformer decoder to generate variable-length offloading sequences $\mathbf{a}_{1:\tau}$, inherently enforcing task priorities, server availability and long-horizon dependencies. Specifically, the causal Transformer in QTTO refers to an autoregressive decoder that uses a strictly lower-triangular self-attention mask to ensure that the representation at step j depends only on the prefix $\mathbf{a}_{1:j-1}$, preventing information leakage from future decisions. The causal mask prevents each decoding step from attending to future decisions, thereby preserving the autoregressive property and ensuring that the current decision is generated only based on available historical decisions [14], [32]. In addition, a dynamic feasibility mask is applied at each decoding step to invalidate actions corresponding to already occupied or infeasible servers. A TokenLearner module compresses heterogeneous states into compact, learnable tokens, while dueling heads and n-step returns enhance training stability and credit assignment.

1) State Tokenization and Encoding: To handle variable input dimensions and high heterogeneity, QTTO applies linear projection layer as the **TokenLearner**. This tokenizer maps heterogeneous features S_t into G learnable tokens in a unified embedding space of dimension d, ensuring that the self-attention mechanism can compute meaningful dependencies across different physical scales, depicted as the Feature Extraction module in Fig. 3 and denoted by

$$\mathbf{Z} = \text{TokenLearner}(S_t) \in \mathbb{R}^{B\times G\times d}, \tag{22}$$

followed by a L-layer Transformer encoder with the Encoding Tensor module and FlashAttention by

$$\mathbf{H} = \text{TransformerEncoder}(\mathbf{Z}) \in \mathbb{R}^{B\times G\times d}. \tag{23}$$

This design enables cross-attention between tasks and servers while maintaining $O((M+K)^2 d)$ complexity, which is significantly lower than the M^K brute-force enumeration.

2) Autoregressive Q-Value Estimation: QTTO estimates the return-to-go for partial action sequences using a causal decoder, represented by the purple Transformer Blocks in Fig. 3. Beginning with a start-of-sequence (SOS) token derived from $\mathbf{H}$, prior actions are embedded via learned bins $\mathbf{W}_a \in \mathbb{R}^{|\mathcal{A}|\times d}$, depicted as

$$Q_\theta(S_t, \mathbf{a}_{1:j}) = \text{Decoder}_\theta(\mathbf{a}_{1:j}; \mathbf{H}), \tag{24}$$

where the decoder corresponding to the pink Attention Architecture block applies causal self-attention to condition exclusively on prior decisions, cross-attention to encoder outputs $\mathbf{H}$ and dynamic feasibility masking to invalidate reassignment to occupied servers. Concretely, the causal masking follows a standard autoregressive formulation with a lower-triangular attention matrix, while cross-attention remains unmasked to allow full access to the encoded state tokens $\mathbf{H}$, thereby enabling task-server dependency modeling without violating temporal causality [14], [32].

The next-action distribution is

$$\mathbf{q}_{j+1}(a) = \text{softmax}\left(\mathbf{h}_j^{\text{dec}} \mathbf{W}_a^\top + \mathbf{m}_{j+1}\right). \tag{25}$$

TABLE III
KEY QTTO ALGORITHMIC SETTINGS

Category	Key Design
Optimizer	AdamW
Gradient stabilization	Gradient clipping
Return estimation	n-step return-to-go
Value decomposition	Dueling heads ($V + A - \mathbb{E}[A]$)
Target network	EMA update
State encoding	TokenLearner-based compression
Inference	Greedy autoregressive decoding

Here, $\mathbf{m}_{j+1}$ encodes the feasibility mask, where infeasible actions are assigned $-\infty$ to ensure zero probability after softmax, thereby integrating architectural masking and decision constraints in a unified manner. At inference, greedy decoding selects $a^{(j)} = \arg\max(\mathbf{q}_j)$ until task completion or early termination, yielding conflict-free plans with quadratic inference complexity.

This formulation ensures priority preservation, conflict-free allocation and adaptive sequencing under runtime dynamics.

3) Dataset Replay and Training: To enable the model to learn the complex, sequence-aware Q-function, a training process based on dataset replay is introduced.

The dataset $\mathcal{D} = \{(S_t, \mathbf{a}^*_{1:\tau}, \mathbf{R}^*_{1:\tau})\}$ is collected through QTTO exploration and stored in the Replay Dataset module shown in Fig. 3. Unlike DQN's single-step transitions, QTTO trains on full sequences based on n-step return-to-go, denoted as

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathcal{D}} \left[\sum_{j=1}^{\tau^*} \left\| Q_\theta(S_t, \mathbf{a}^*_{1:j}) - \hat{R}^{(n)}_{j:\tau^*} \right\|^2 \right], \tag{26}$$

where

$$\hat{R}^{(n)}_{j:\tau^*} = \sum_{k=j}^{\min(j+n-1,\tau^*)} \gamma^{k-j} r^*_k + \gamma^n Q_\theta(S_{t+j+n-1}, \mathbf{a}^*_{j+n:\tau^*}). \tag{27}$$

The model stability is enhanced through **dueling heads** with $Q = V(S_t) + (A - \mathbb{E}[A])$ for improved value decomposition, an **exponential moving average (EMA) target network** for consistent learning targets and **conservative regularization** to mitigate overestimation.

Its parameters are optimized using AdamW with gradient clipping and the key algorithmic settings are summarized in Table III.

4) Rationale for Transformer Architecture Design: The architecture is designed to balance representational power, tractable decoding and deployment feasibility under the stringent latency and resource constraints of MEC systems. We employ learned positional embeddings to represent the fixed identity of edge servers and the relative priority of tasks. Although sinusoidal encodings are attractive for extrapolation to unseen sequence lengths, this advantage is less critical in our MEC setting, where the number of servers M and the maximum task batch size K are fixed deployment-time parameters. Under this assumption, learned embeddings are a practical choice because they allow each positional index to correspond to a fixed identity in the network, while task tokens can learn representations that reflect ordinal priority and contextual importance.

Algorithm 1: QTTO Autoregressive Inference.

Input: State S_t, model θ, action space $\mathcal{A} = \{0, 1, \ldots, M\}$
Output: Offloading sequence $\mathbf{a}_{1:\tau}$
$\mathbf{H} \leftarrow$ Encoder(TokenLearner(S_t))
Initialize: tokens $\leftarrow$ [SOS($\mathbf{H}$)], $\mathbf{a} \leftarrow []$, used_servers $\leftarrow \emptyset$
for $j = 1$ to K **do**
  $\mathbf{h}_j \leftarrow \text{Decoder}_\theta(\text{tokens}; \mathbf{H})$
  $\mathbf{q}_j \leftarrow \text{softmax}(\mathbf{h}_j \mathbf{W}_a^\top + \mathbf{m}_j)$ ($\mathbf{m}_j(m) = -\infty$ if $m \in$ used_servers)
  $a^{(j)} \leftarrow \arg\max \mathbf{q}_j$
  Append: $\mathbf{a} \leftarrow \mathbf{a} \cup \{a^{(j)}\}$, tokens $\leftarrow$ tokens $\cup \{\text{Embed}(a^{(j)})\}$
  Update used_servers $\leftarrow$ used_servers $\cup \{a^{(j)}\}$ if $a^{(j)} > 0$
  if no tasks remain: **break**
end for
Return: $\mathbf{a}_{1:\tau}$

Furthermore, we adopt an encoder-decoder architecture to better handle the combinatorial action space in task offloading. The encoder first extracts global system-level representations from the observed state, while the causal decoder generates offloading decisions in an autoregressive manner so that each decision is conditioned only on previously assigned tasks. Compared with a non-autoregressive or encoder-only design, this factorized decoding strategy enables each offloading decision to be conditioned on previously generated decisions, providing a practical way to model inter-task dependencies and sequential decision constraints during inference [12], [14], [32]. The causal mask prevents each decoding step from attending to future decisions, thereby preserving the autoregressive property and ensuring that the current decision is generated only based on available historical decisions [14], [32]. Overall, this architecture offers a practical balance among representational capacity, decoding tractability and deployment feasibility for complex MEC environments.

The ablation study in Section VI-D evaluates the contribution of this architecture.

Algorithm 1 summarizes the QTTO autoregressive inference procedure.

By transforming MEC offloading into sequence prediction with causal constraints, QTTO achieves exact priority enforcement and scalable inference. In the autoregressive setting, the Q-value computation considers the sequence of previously selected actions, enabling the model to capture task dependencies and make more informed offloading decisions.

C. Federated Learning Integration

To preserve data locality while learning a shared task-oriented policy, FQTTO integrates the QTTO framework with a FL paradigm. This approach, based on the Federated Averaging (FedAvg) algorithm, enables collaborative training of a global

offloading policy without exchanging raw, privacy-sensitive user data [35]. In this work, we adopt a centralized aggregation architecture because it provides a relatively stable and communication-efficient mechanism for synchronizing model updates across heterogeneous devices, which is important under non-IID data distributions and dynamic MEC conditions [36].

Specifically, compared with fully decentralized FL, the centralized design offers a simpler and typically more stable aggregation mechanism, while weighted aggregation helps maintain a coherent global training objective. In our setting, the server-based FL design follows the commonly used FedAvg-style paradigm, where local model updates are aggregated at a central server to form a shared global model. Weighted aggregation is also consistent with FedAvg-style training, in which client contributions are commonly weighted according to local data sizes to maintain a coherent global optimization objective [35], [36]. This is important for the QTTO model, where sequence-level Q-value estimation can be sensitive to parameter drift and therefore benefits from consistent global coordination during training.

The centralized coordinator can be naturally deployed at an edge server, reducing synchronization complexity while preserving data locality by exchanging model updates rather than raw data. Although parameter sharing may still incur residual privacy risks, this design provides a practical trade-off among privacy preservation, training stability and system efficiency.

Instead of a single, centralized dataset $\mathcal{D}$, each participating user device $n \in \mathcal{N}$ maintains its own local QTTO model parameters θ_n and a local experience replay buffer $\mathcal{D}_n$. This buffer stores trajectories collected during interaction with its local MEC environment.

The federated training protocol consists of three primary phases.

1) Local Exploration and Data Collection: In each communication round r, each device n first acts as an agent to interact with its environment. Using its current local model $\theta_n(r)$, it performs autoregressive inference, as detailed in Algorithm 1, to make offloading decisions. The resulting trajectories are collected and stored in its local replay buffer $\mathcal{D}_n$. This design is inherently flexible in that $\mathcal{D}_n$ can be pre-populated with a static offline dataset generated from simulations or system logs (offline pre-training), dynamically and continuously populated by the agent's real-time interactions (online learning), or constructed from a combination of both.

2) Local Model Training: After the exploration phase or when a predefined training condition is met, each device n performs E local training epochs. In each epoch, it samples a mini-batch of trajectories from its local buffer $\mathcal{D}_n$ and applies the core QTTO training objective. Specifically, it computes the n-step return-to-go targets $\hat{R}^{(n)}_{j:\tau}$ and updates its local model parameters θ_n by minimizing the dueling Q-learning loss via stochastic gradient descent, represented as

$$\theta_n \leftarrow \arg\min_{\theta} \mathbb{E}_{(S,\mathbf{a}_{1:\tau},\mathbf{R}_{1:\tau})\sim\mathcal{D}_n}\left[\sum_{j=1}^{\tau}\left\|Q_\theta(S,\mathbf{a}_{1:j}) - \hat{R}^{(n)}_{j:\tau}\right\|^2\right]. \tag{28}$$

Algorithm 2: FQTTO Federated Training and Inference.

Initialize: Global model $\theta_g(0)$, local models $\theta_n(0) = \theta_g(0)\ \forall n \in \mathcal{N}$
for communication round $r = 0$ to $R-1$ **do**
  for device $n \in \mathcal{N}$ **in parallel do**
    Generate local dataset $\mathcal{D}_n$ via QTTO exploration
    Train $\theta_n(r)$ on $\mathcal{D}_n$ using n-step dueling return-to-go loss (26) for Φ local iterations to obtain $\theta_n(r+1)$
    Upload updated parameters $\theta_n(r+1)$ to coordinator
  end for
  Aggregate global model:

$$\theta_g(r+1) \leftarrow \sum_{n=1}^{N} \frac{|\mathcal{D}_n|}{\sum_{k=1}^{N}|\mathcal{D}_k|}\theta_n(r+1)$$

  Broadcast: $\theta_n(r+1) \leftarrow \theta_g(r+1)\ \forall n$
end for
Deployment Phase:
Use final global model $\theta_g(R)$ with QTTO autoregressive inference in Algorithm 1 for real-time, priority-aware task offloading

This local training allows θ_n to adapt to the specific dynamics and task characteristics of device n's own environment.

3) Global Model Aggregation and Distribution: Following the local training phase, each device n transmits only its updated model parameters $\theta_n(r+1)$ to a central edge coordinator, such as an aggregation server. The coordinator performs a weighted average of the parameters from all N devices to compute the new global model $\theta_g(r+1)$, denoted by

$$\theta_g(r+1) = \sum_{n=1}^{N} w_n \theta_n(r+1), \quad w_n = \frac{|\mathcal{D}_n|}{\sum_{k=1}^{N}|\mathcal{D}_k|}. \tag{29}$$

The weighting w_n by the size of the local dataset $|\mathcal{D}_n|$ helps mitigate the impact of data imbalance and client drift in non-independent and non-identically distributed (non-IID) settings.

Finally, the coordinator broadcasts the updated global model $\theta_g(r+1)$ back to all devices, which then set their local models for the next round of exploration and training by

$$\theta_n(r+1) = \theta_g(r+1), \quad \forall n \in \mathcal{N}. \tag{30}$$

Combining autoregressive sequence modeling with federated updates, FQTTO enables privacy-aware learning across heterogeneous MEC devices, achieving scalable and adaptive task offloading.

D. FQTTO Training and Deployment

The FQTTO algorithm, which orchestrates the federated training of the QTTO policy, is formalized in Algorithm 2. This subsection discusses the transition from federated training to deployment and the computational characteristics of the overall framework.

The federated training process is executed until the global model θ_g converges or a predefined number of communication

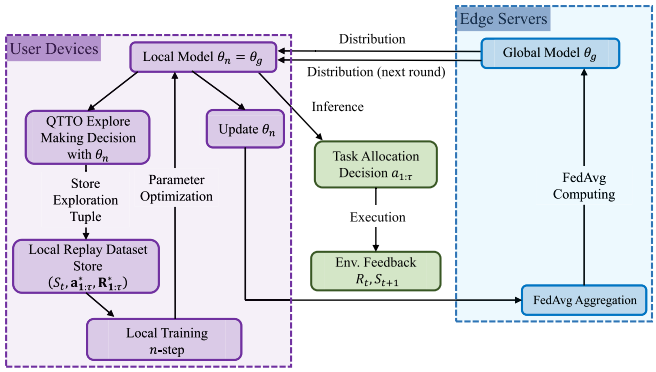


Fig. 4. FQTTO federated learning pipeline. Local devices generate sequence datasets via QTTO exploration, perform n-step training on local replay buffers, upload model updates for weighted FedAvg aggregation and receive the global model for synchronized inference.

rounds R is completed. Once training is terminated, the final global model $\theta_g(R)$ is deployed to all user devices. This deployed model offers two distinct operational modes, providing flexibility for real-world MEC systems. For environments with relatively stable conditions, the model parameters can be frozen for static inference, where devices use $\theta_g(R)$ purely for high-speed, low-overhead inference. For dynamic environments, the model can operate in an online fine-tuning mode, so that the transition from training to deployment remains smooth and the narrative from algorithm design to practical use stays continuous. Here, $\theta_g(R)$ serves as a pre-trained base and devices continue to collect new experiences in their local replay buffers $\mathcal{D}_n$ to perform periodic local training. This allows agents to adapt to long-term concept drift, such as changes in network topology, server capabilities or task distributions in their local environments, ensuring sustained performance. Sporadic lower-frequency aggregation rounds can be performed to consolidate these incremental model updates.

The computational complexity of FQTTO is managed across its phases. Let P be the number of model parameters, N the number of devices, $|\mathcal{D}_n|$ the average local dataset size, Φ the number of local training iterations, R the communication rounds, K the maximum task sequence length and d the embedding dimension. The local training phase dominates the federated process, with a complexity of $O(R \times N \times |\mathcal{D}_n| \times \Phi \times P)$, though this is fully parallelized across N devices. Communication is efficient, requiring only $O(N \times P)$ for aggregation and broadcast, independent of data size. Critically, real-time inference is highly efficient, with a complexity of $O(K^2 \tilde{} d)$ per decision via greedy decoding, making it suitable for low-latency MEC environments.

Fig. 4 illustrates the end-to-end FQTTO workflow.

VI. Simulation Experiments

In this section, we evaluate the performance of the FQTTO algorithm through extensive simulation experiments, comprising comparison experiments, extension experiments and ablation experiments. The evaluation focuses on key metrics, including task delay, energy consumption, load balance and task completion rate, under diverse network conditions and device configurations.

A. Basic Environment Settings

1) Dataset and Network Description: The dataset comprises 6000 tasks with heterogeneous characteristics, designed to emulate realistic edge computing workloads inspired by Alibaba Cluster Trace analyses. It includes 20% delay-sensitive and 80% computation-intensive tasks, reflecting long-tail distributions. Task attributes are primarily generated using Pareto distributions ($\alpha \approx 1.5$, where α denotes the shape parameter and the scale parameter x_m is set to 1 for normalized sampling) for skewed, heavy-tailed behaviors, with additional uniform distributions for specific parameters. Specifically, we first sample normalized variables $x \sim \text{Pareto}(x_m{=}1, \alpha)$ and then apply linear scaling and truncation to map them into practical ranges with physical units.

For delay-sensitive tasks:

- *Data size:* Pareto-distributed, scaled to [0.1, ∼2] MB, obtained via linear scaling of normalized Pareto samples with upper-tail truncation.
- *Computational complexity:* Pareto-distributed, [1, ∼10] cycle/bit, biased lower through scaling factors that emphasize smaller sampled values.
- *Deadline:* Uniformly distributed in [0.05, 0.5] s.
- *Memory needs:* Pareto-distributed, scaled to [10, ∼50] MB and calculated from normalized Pareto samples with truncation for stability.

For computation-intensive tasks:

- *Data size:* Pareto-distributed, scaled to [1.0, ∼20] MB using a larger scaling factor from normalized Pareto samples.
- *Computational complexity:* Pareto-distributed, [1, 10] cycle/bit, biased higher by applying larger scaling factors to emphasize higher computational demand.
- *Deadline:* Uniformly distributed in [0.5, ∼5] s.
- *Memory needs:* Pareto-distributed, scaled to [100, ∼300] MB, derived from normalized Pareto samples with upper-tail truncation.

Additional attributes include sequential task IDs and their home users.

The users have heterogeneous devices, with 30% high-end and 70% standard. Transmission power is [0.05, 0.5] W and distance to edge is [10, 1000] meters.

The edge computing infrastructure consists of 16 heterogeneous edge servers with CPU frequencies ranging from 2.0 to 4.5 GHz and a 40 MHz wireless bandwidth with dynamic channel conditions, reflecting standard configurations for networks and local learning every 5 epochs for collaborative training.

The critical simulation parameters are detailed in Table IV, tuned via grid search over ranges such as learning rate [1e-5,1e-3] and discount factor [0.95, 0.995]. The remaining parameters are fixed, such as the reward coefficients in (20) set to $\alpha = 0.4$ for delay, $\beta = 0.3$ for energy, $\gamma = 0.3$ for load balance.

These settings establish a realistic and reproducible environment protocol that remains fixed in most cases across the subsequent experiments, thereby guaranteeing an equitable

TABLE IV
BASIC ENVIRONMENT PARAMETERS

Parameter	Value
Number of Tasks	6000
Data Size	[0.1, ∼20] MB
Complexity	[0.5, ∼15] cycles/bit
Deadline	[0.05, 5.0] s
Bandwidth	40 MHz
Base Channel Gain	0.1
Edge Servers	16
Users	64
White Noise Power	-174 dBm/Hz
Discount Factor	0.993
Embedding Dims	256
Learning Rate	$1e$-4
Transformer Layers	4
Transformer Attention Heads	16

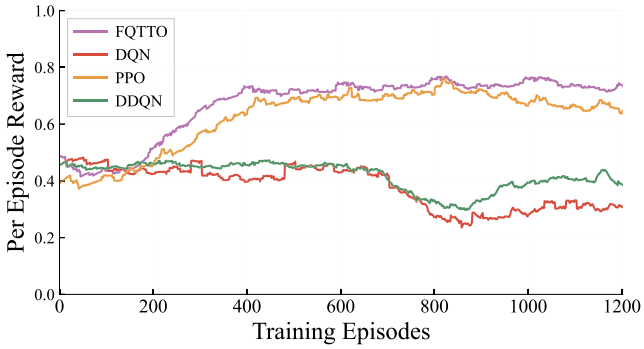


Fig. 5. Reward comparison of the algorithms.

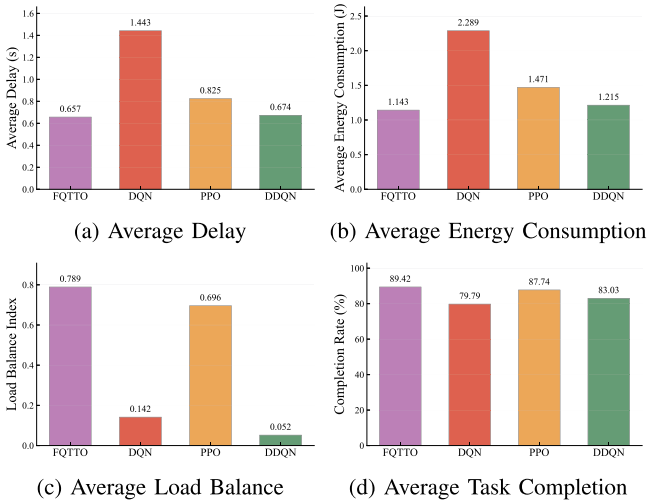


Fig. 6. Performance comparison of the algorithms.

comparison between FQTTO and the baselines under a broad spectrum of MEC scenarios.

2) Benchmark: We introduced the following algorithms for comparison: FQTTO, Deep Q-Network (DQN), Double Deep Q-Network (DDQN) and Proximal Policy Optimization (PPO). FQTTO is trained under the proposed federated learning framework, whereas DQN, DDQN and PPO are used as standard centralized RL baselines for reference. These algorithms are evaluated based on metrics such as mean time from task submission to completion, energy usage of each task, load balance of task measured by normalized standard deviation and percentage of tasks completed within their deadlines.

B. Comparison Experiments

To evaluate the performance of the FQTTO algorithm, we conducted extensive experiments comparing it against DQN, DDQN and PPO.

1) Convergence: To assess the convergence characteristics of the proposed algorithm, a detailed comparative analysis was conducted against baselines. Fig. 5 presents the per episode reward trajectories across 1200 training episodes, providing a quantitative measure of performance stability and convergence efficiency.

The results indicate that the proposed FQTTO algorithm exhibits the most robust convergence, achieving a per episode reward approaching 0.75 after approximately 400 episodes. This represents an improvement over PPO, DQN and DDQN, which stabilize at lower reward levels of approximately 0.7, 0.4 and 0.3, respectively. Notably, FQTTO demonstrates a steeper learning curve and sustains higher reward values throughout the training process, suggesting enhanced adaptability and optimization capability. In contrast, PPO displays moderate convergence with fluctuations, while DDQN and DQN exhibit the least effective convergence, maintaining a consistently lower reward. These findings underscore the superior convergence properties of FQTTO, positioning it as a promising advancement in the domain of RL algorithms.

2) Metrics and Comparison Analysis: The evaluation focuses on key performance metrics: average delay, average energy consumption, load balance index and task completion rate. These metrics are critical for assessing the effectiveness of task offloading algorithms in dynamic and heterogeneous computing environments.

Fig. 6 presents a detailed comparison of the algorithms across the metrics.

As illustrated in Fig. 6(a), FQTTO achieves the lowest average response time of 0.657 s, outperforming DQN (1.443 s), PPO (0.825 s) and DDQN (0.674 s). This represents a reduction of 54.4% relative to DQN, 20.3% relative to PPO and 2.5% relative to DDQN, yielding an average delay improvement exceeding 20.4% across the baselines. FQTTO effectively captures temporal dependencies in task queues and network states, enabling more adaptive offloading decisions compared to the value-based approximations in DQN and DDQN or the policy-gradient approach in PPO.

Fig. 6(b) presents the average energy consumption results, where FQTTO records the minimal value of 1.143 J, compared to 2.289 J for DQN, 1.471 J for PPO and 1.215 J for DDQN. This corresponds to energy savings of 50.0% on DQN, 22.3% on PPO and 5.9% on DDQN, with an average reduction greater than 22.3%. The baselines exhibit higher energy overheads due to less optimized handling of heterogeneous device capabilities and dynamic energy constraints.

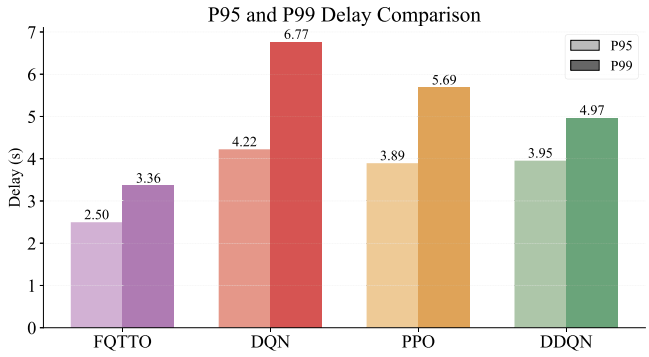


Fig. 7. Tail-latency comparison of FQTTO and baseline methods in terms of P95 and P99 delay metrics.

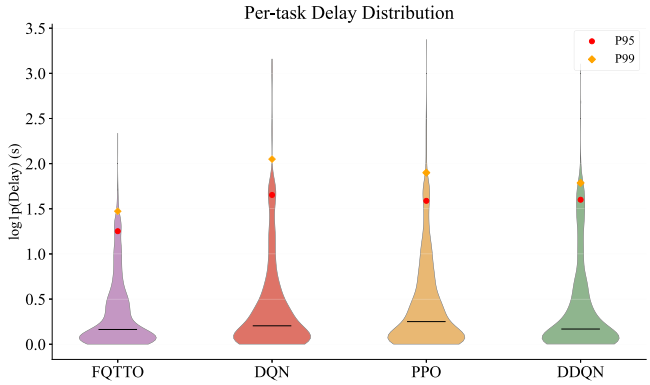


Fig. 8. Violin plot of the delay distribution for FQTTO and baseline methods.

Fig. 6(c) shows that FQTTO attains the highest load balance index of 0.789, significantly surpassing DQN (0.142), PPO (0.696) and DDQN (0.052). This enhancement, showing a 13.3% improvement over the closest baseline (PPO), underscores FQTTO's ability to equitably distribute tasks across edge servers by leveraging attention mechanisms in the Transformer architecture to prioritize load-aware offloading. The baselines, lacking such sophisticated dependency modeling, result in uneven resource utilization, particularly under high-variability conditions.

Fig. 6(d) highlights the task completion rate, with FQTTO achieving 89.42%, compared to 79.79% for DQN, 87.74% for PPO and 83.03% for DDQN. This reflects improvements ranging from 1.9% to 12.1% across the baselines, with a minimum gain of 1.9% over PPO and a maximum of 12.1% over DQN, which ensures the QoS of users.

Overall, these results indicate that FQTTO achieves a favorable performance profile for the multi-objective trade-offs considered in our MEC setting. The framework's robustness is reflected in its overall performance across the evaluated metrics, supported by the combination of FL-based training and Transformer-based decision modeling for complex dynamic states. Notably, the Transformer encoder enables structured representation of sequential task states and is better able than the MLP-based baseline to capture long-range dependencies in the evaluated setting.

3) Tail-Latency Analysis: To further evaluate the proposed FQTTO under heavy-tailed workloads, we analyze tail latency using the P95 and P99 metrics. As shown in Fig. 7, FQTTO achieves a P99 delay of 3.36 s, which is 50.3% lower than that of DQN. This result indicates that FQTTO achieves better tail-latency performance, where urgent tasks are more likely to be affected by workload dynamics and resource contention.

Fig. 8 further shows the full delay distribution across tasks. Compared with the evaluated baseline methods, FQTTO exhibits a tighter delay distribution and a less pronounced long-tail effect. These results are consistent with the view that the proposed architecture helps the policy better handle complex task dependencies and workload variability, which is beneficial for latency-sensitive tasks in dynamic MEC environments.

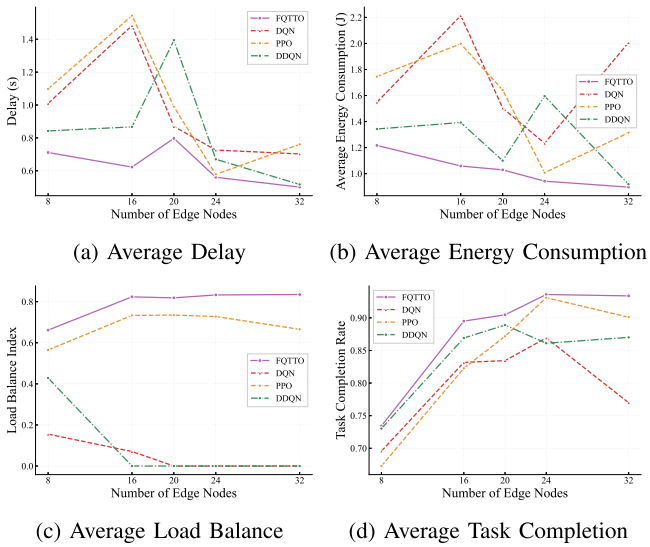
C. Extension Experiments

To assess FQTTO's scalability and adaptability, we conducted experiments under varying user and task conditions and device heterogeneity.

1) Evaluating Strategy:

- *Node Scalability:* We varied the number of edge servers from 8 to 32, while keeping other parameters fixed.
- *User Dynamics:* We simulated dynamic user scenarios by varying the number of active user devices from 24 to 120, with random task arrival rates, while keeping other parameters fixed.
- *Dataset Diversity:* We evaluated FQTTO across four distinct task datasets to capture heterogeneity in real-world workloads. These datasets were generated using Pareto distributions to simulate long-tail effects observed in empirical traces. The types include
  - *Hybrid:* A balanced mix with 20% delay-sensitive tasks (low complexity, short deadlines) and 80% compute-intensive tasks (higher complexity, longer deadlines), representing general-purpose edge workloads.
  - *Sensitive:* All tasks are delay-sensitive, with low complexity (1–10 cycles/bit), small data sizes (0.1–0.8 MB) and tight deadlines (0.01–0.3 s), simulating real-time applications like AR or IoV.
  - *Dense:* All tasks are compute-intensive, with higher complexity (5–15 cycles/bit), larger data sizes (5–25 MB) and extended deadlines (1–10 s), modeling data-heavy scenarios such as IIoT sensor processing.
  - *Burst:* Primarily normal tasks (similar to Hybrid), but with 10% anomalous bursts featuring extreme data sizes up to 60 MB or complexity up to 1.5 times the normal maximum, mimicking sudden traffic spikes in networks based on CAIDA datasets.
- *Bandwidth Variant:* We tested under varying wireless bandwidths (10 MHz, 30 MHz, 50 MHz, 70 MHz, 100 MHz) to evaluate performance in constrained and high-capacity networks, while keeping other parameters fixed.

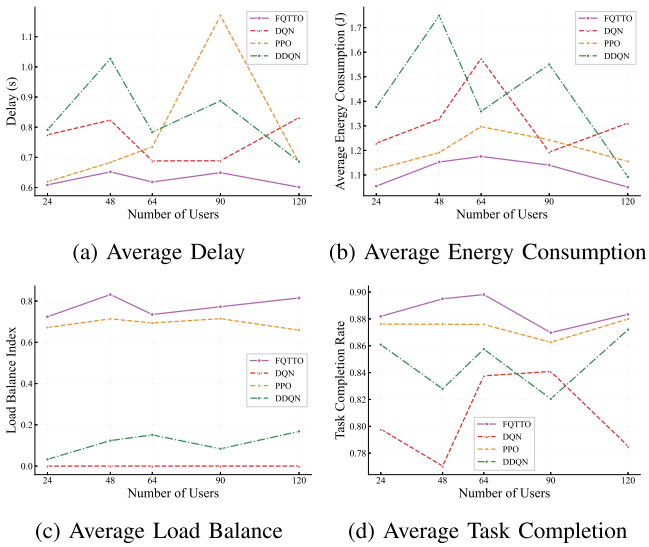
2) Extension Results and Analysis: To further assess the scalability and robustness of FQTTO in diverse edge computing scenarios, we conducted extension experiments by varying key



(a) Average Delay (b) Average Energy Consumption

(c) Average Load Balance (d) Average Task Completion

Fig. 9. Extension results of varying edge servers.



(a) Average Delay (b) Average Energy Consumption

(c) Average Load Balance (d) Average Task Completion

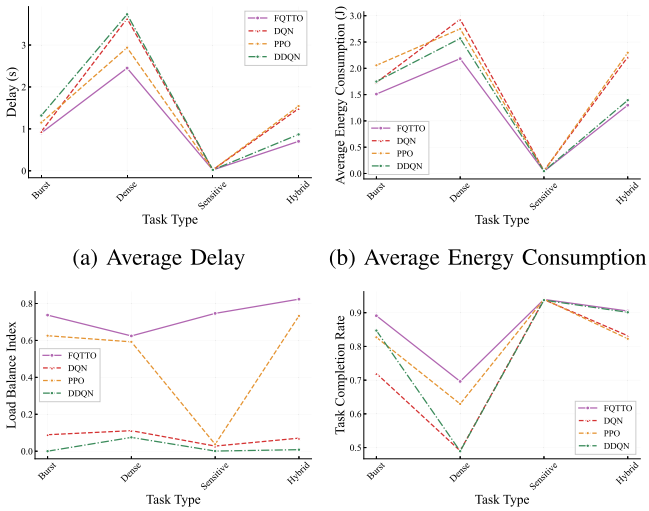
Fig. 10. Extension results of varying users.

system parameters, described in the evaluating strategy above. Performance metrics, including task completion rate, energy consumption, load balance and delay, were measured across multiple runs, with the baselines (DQN, PPO and DDQN) for comparison.

We evaluated FQTTO's performance by scaling the number of edge servers from 8 to 32, simulating growing network density in heterogeneous MEC environments. As shown in Fig. 9(a), FQTTO consistently minimizes average delay, outperforming baselines that exhibit greater variability and elevated values. Energy consumption follows a similar trend in Fig. 9(b), where FQTTO achieves substantial reductions, yielding notable savings over baselines. The load balance index, depicted in Fig. 9(c), demonstrates FQTTO's superior resource distribution, maintaining equilibrium even at higher node counts and surpassing baselines. Finally, as shown in Fig. 9(d), FQTTO achieves the highest task completion rate, reflecting a 27.1% improvement as nodes increase, outperforming baselines by an average of 13.4%.

To examine FQTTO's resilience under increasing user load, we varied the number of users from 24 to 120. Fig. 10(a) reveals that FQTTO manages delay escalation effectively, remaining markedly lower than baselines despite increased load. Energy consumption rises modestly for FQTTO in Fig. 10(b), contrasting with sharper increases in baselines. The load balance index decreases by 15.8% for FQTTO in Fig. 10(c), yet remains over 20% higher than baselines. Fig. 10(d) depicts the task completion rate, where FQTTO sustains high performance with only a 6.6% drop, outperforming baselines by 8–15% on average.

Adaptability to heterogeneous workload patterns was tested using four task types: Burst, Dense, Sensitive and Hybrid. Fig. 11(a) shows that FQTTO achieves the lowest average delay, with a reduction of 24.2% versus baselines, due to its effective handling of dynamic workloads through Transformer-based state modeling. The average energy consumption is minimized by FQTTO, with an average 20.7% savings vs. baselines, as illustrated in Fig. 11(b), reflecting optimized task offloading that



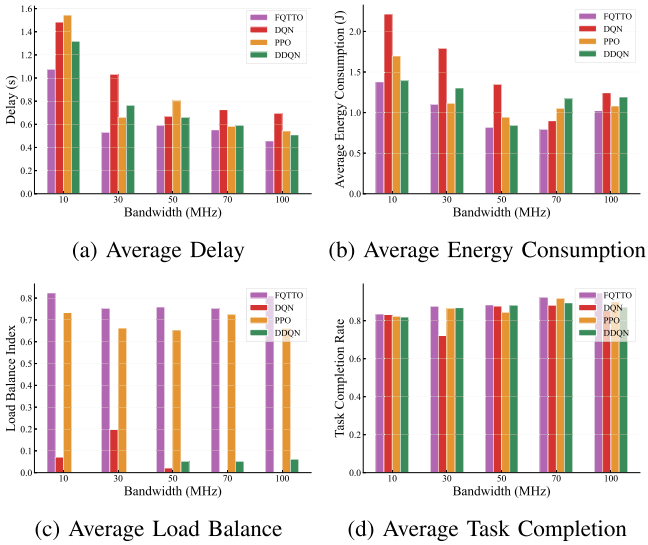
(a) Average Delay (b) Average Energy Consumption

(c) Average Load Balance (d) Average Task Completion

Fig. 11. Extension results of varying datasets.

reduces unnecessary energy expenditure. Load balance index achieves optimal utilization in Fig. 11(c), while Fig. 11(d) shows task completion rates, where FQTTO averages 12.5% higher than baselines across types, highlighting its robustness in maintaining high completion rates even under challenging conditions.

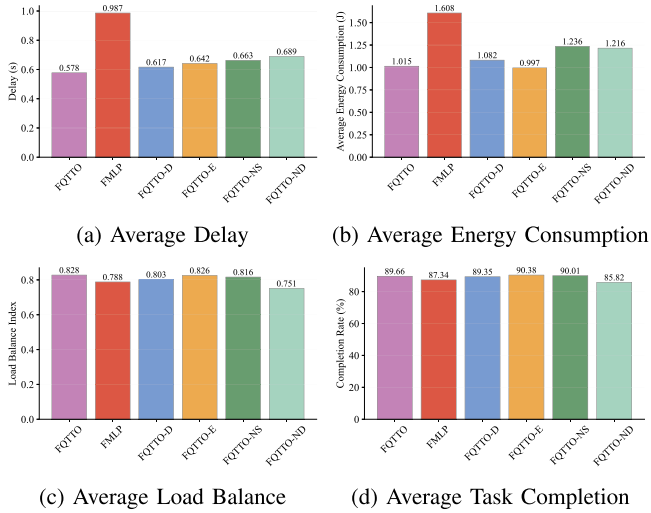
Sensitivity to network communicating conditions was explored by varying bandwidth from 10 to 100 MHz. The average delay for FQTTO diminishes sharply in Fig. 12(a), consistently outperforming baselines, the average energy consumption declines in Fig. 12(b), load balance stabilizes in Fig. 12(c) and task completion rate improves consistently in Fig. 12(d), surpassing baselines across the varying bandwidths.



(a) Average Delay (b) Average Energy Consumption

(c) Average Load Balance (d) Average Task Completion

Fig. 12. Extension results of varying bandwidths.



(a) Average Delay (b) Average Energy Consumption

(c) Average Load Balance (d) Average Task Completion

Fig. 13. Performance comparison between ablated variants and the complete framework.

In summary, these extension experiments across varying nodes, users, tasks, types and bandwidths affirm FQTTO's robustness and scalability in dynamic and heterogeneous MEC environments. Compared to baselines, FQTTO consistently outperforms in the key metrics. These gains are realized while preserving data locality through FL, thereby ensuring privacy guarantees without compromising multi-objective QoS.

D. Ablation Experiments

To complement the baseline comparison, we finally isolate the contributions of key architectural and learning components in FQTTO through controlled ablation studies. To understand the contributions of FQTTO's components, we performed ablation studies by systematically removing key elements while keeping others.

1) Ablated Variants: There are five ablated variants, depicted as below.

- *FMLP:* Replace the Transformer component with MLP equipped with a Dueling architecture.
- *FQTTO-D:* Single-objective optimization for delay only.
- *FQTTO-E:* Single-objective optimization for energy only.
- *FQTTO-NS:* Without n-step learning.
- *FQTTO-ND:* Without dueling heads.

Among these, FMLP serves as a non-Transformer baseline for representation learning, while FQTTO-NS removes the multi-step return mechanism. Together, they provide a direct experimental proxy for evaluating Transformer-based and multi-step frameworks within a controlled setting.

2) Ablation Results: To evaluate the efficiency of the components, we test them across four dimensions, including delay, average energy consumption, load balance and task completion rate on average. Fig. 13 shows the results of the detailed ablation study, quantifying the contribution of each component.

As illustrated in Fig. 13(a), FQTTO delivered the lowest average delay of 0.578 s, a 41.4% improvement over FMLP (0.987 s). Variants such as FQTTO-D (0.617 s) and FQTTO-ND (0.689 s) exhibited increased delays, confirming the necessity of dynamic modeling for adaptive, low-delay performance in volatile environments.

Fig. 13(b) reveals that FQTTO attains an energy consumption of 1.015 J, representing a 36.9% reduction compared to FMLP (1.608 J). This highlights the Transformer's superior capacity to model complex dependencies, leading to more energy-efficient offloading decisions. Notably, FQTTO-E yields a marginally lower energy consumption (0.997 J), likely due to relaxed optimization trade-offs, but this comes at the expense of other metrics, as evidenced elsewhere.

In terms of load balance, Fig. 13(c) demonstrates that FQTTO achieves the highest index of 0.828, outperforming all variants, with FQTTO-ND recording the lowest value (0.751). This improvement of 10.3% over FQTTO-ND emphasizes the queuing mechanism's contribution to equitable resource distribution across edge servers, mitigating hotspots and improving system robustness.

Finally, Fig. 13(d) shows that FQTTO achieves a task completion rate of 89.66%, surpassing FMLP (87.34%) and FQTTO-ND (85.82%), the latter exhibiting the most substantial degradation. This underscores the critical role of the dueling-head component in prioritizing tasks under resource constraints, enhancing overall system reliability by approximately 4.5% relative to FQTTO-ND.

3) Component Contribution Analysis: The ablation study reveals several key insights into the contributions of individual components within the FQTTO framework, quantifying their impact on system performance in heterogeneous and dynamic edge networks.

- *Transformer Superiority:* The comparison with FMLP, which employs a traditional MLP-based architecture, demonstrates that the Transformer provides 41.4% better delay performance and 36.9% energy efficiency improvement over traditional MLP. This performance gap

provides empirical support for the use of a Transformer-based representation in our framework. Compared with the MLP-based variant, the Transformer-based model better captures the heterogeneous interactions between servers and task flows in our evaluated setting, which supports our architectural choice for complex MEC environments.

- *Multi-Objective Optimization:* Single-objective variants underscore the importance of balancing multiple objectives. FQTTO-Eoptimizes only energy, achieves lower energy consumption by 1.8%, but higher delay by 11.1%, while FQTTO-Doptimizes only delay, results in higher energy by 6.7%. These trends highlight the trade-offs inherent in single-objective focus and affirm the necessity of integrated multi-objective reward design for holistic optimization.
- *N-Step Learning Contribution:* Removing n-step learning increases delay by 14.7% and energy consumption by 21.8%, highlighting its importance for long-term reward optimization and improved sample efficiency in dynamic environments. This result indicates the effectiveness of multi-step learning compared to single-step updates, addressing the need for multi-step evaluation.
- *Dueling Heads Benefits:* The dueling heads contribute to 19.2% delay improvement and 19.8% energy savings by better separating state value and action advantage estimation, enabling more precise Q-value approximations and enhanced decision-making stability.

Collectively, these findings affirm that the integrated components of FQTTO, including Transformer architecture, dueling architecture and n-step learning, operate synergistically to optimize multi-objective performance, validating the framework's design in heterogeneous and dynamic edge computing scenarios.

VII. Conclusion and Future Work

In this paper, we proposed FQTTO, which integrates FL and Transformer-based Q-learning to address task offloading challenges in MEC, achieving significant improvements in energy efficiency, delay and load balance. The advantages position FQTTO as a promising solution for real-world MEC deployments, particularly in scenarios with heterogeneous devices, dynamic network conditions and privacy requirements.

Future work will gather larger real-world datasets from diverse MEC deployments, to validate FQTTO's performance in practical settings. We also plan to conduct comprehensive comparative studies with additional algorithms, in order to further demonstrate the relative strengths and potential limitations of FQTTO under varying conditions. Furthermore, we intend to investigate adaptive learning rate mechanisms in FL to improve model accuracy, ensuring scalability in large-scale edge environments.

References

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

[2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surv. Tuts.*, vol. 19, no. 4, pp. 2322–2358, Fourth Quarter 2017.

[3] Z. Tong, J. Deng, J. Mei, Y. Zhang, and K. Li, "Multi-objective DAG task offloading in MEC environment based on federated DQN with automated hyperparameter optimization," *IEEE Trans. Serv. Comput.*, vol. 17, no. 6, pp. 3999–4012, Nov./Dec. 2024.

[4] H. Zhou, H. Dai, G. Yang, and Y. Xiang, "Robust federated learning for privacy preservation and efficiency in edge computing," *IEEE Trans. Serv. Comput.*, vol. 18, no. 3, pp. 1739–1752, May/Jun. 2025.

[5] P. Wang et al., "Decentralized navigation with heterogeneous federated reinforcement learning for UAV-enabled mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 13621–13638, Dec. 2024.

[6] Z.-Y. Chai, Y.-J. Zhao, and Y.-L. Li, "Multitask computation offloading based on evolutionary multiobjective optimization in industrial Internet of Things," *IEEE Internet Things J.*, vol. 11, no. 9, pp. 15894–15908, May2024.

[7] L. Dai, F. Zeng, H. Kong, J. Cai, H. Jiang, and K. Li, "Throughput-aware cooperative task offloading in dynamic mobile edge computing systems," *IEEE Trans. Mobile Comput.*, vol. 24, no. 12, pp. 13276–13292, Dec. 2025.

[8] J. Mei, L. Dai, Z. Tong, X. Deng, and K. Li, "Throughput-aware dynamic task offloading under resource constraint for MEC with energy harvesting devices," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 3, pp. 3460–3473, Sep. 2023.

[9] W. Y. B. Lim et al., "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surv. Tuts.*, vol. 22, no. 3, pp. 2031–2063, Third Quarter 2020.

[10] E. Parisotto et al., "Stabilizing transformers for reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, pp. 7977–7996, 2020.

[11] C. Wang, X. Chai, S. Peng, Y. Yuan, and G. Li, "Deep reinforcement learning with entropy and attention mechanism for D2D-assisted task offloading in edge computing," *IEEE Trans. Serv. Comput.*, vol. 17, no. 6, pp. 3317–3329, 2024.

[12] L. Chen et al., "Decision transformer: Reinforcement learning via sequence modeling," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 15084–15097.

[13] P. Agarwal et al., "Transformers in reinforcement learning: A survey," 2023, *arXiv:2307.05979*.

[14] Y. Chebotar et al., "Q-transformer: Scalable offline reinforcement learning via autoregressive Q-functions," 2023, *arXiv:2309.10150*.

[15] S. Dong, Y. Xia, and J. Kamruzzaman, "Quantum particle swarm optimization for task offloading in mobile edge computing," *IEEE Trans. Ind. Informat.*, vol. 19, no. 8, pp. 9113–9122, Aug. 2023.

[16] G. Wu et al., "Combining lyapunov optimization with actor–critic networks for privacy-aware IIoT computation offloading," *IEEE Internet Things J.*, vol. 11, no. 10, pp. 17437–17452, May2024.

[17] C. Li, L. Yu, S. Zeng, Y. Zhang, K. Jiang, and S. Wan, "Stackelberg game-based task offloading for joint service caching and resource allocation optimization in UAV-assisted VEC," *ACM Trans. Internet Things*, vol. 6, no. 1, pp. 1–31, 2025.

[18] M. Tang and V. W. S. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 1985–1997, Jun. 2022.

[19] N. Zhao, Z. Ye, Y. Pei, Y.-C. Liang, and D. Niyato, "Multi-agent deep reinforcement learning for task offloading in UAV-assisted mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 6949–6960, Sep. 2022.

[20] H. Zhai, X. Zhou, H. Zhang, and D. Yuan, "Delay minimization in hybrid edge computing networks: A DDQN-based task offloading approach," *IEEE Trans. Veh. Technol.*, vol. 73, no. 10, pp. 15098–15108, Oct. 2024.

[21] J. Liang, H. Xing, F. Wang, and V. K. N. Lau, "Joint task offloading and cache placement for energy-efficient mobile edge computing systems," *IEEE Wireless Commun. Lett.*, vol. 12, no. 4, pp. 694–698, Apr. 2023.

[22] W. Zhao et al., "DRL connects lyapunov in delay and stability optimization for offloading proactive sensing tasks of RSUs," *IEEE Trans. Mobile Comput.*, vol. 23, no. 7, pp. 7969–7982, Jul. 2024.

[23] A. Mondal, D. Mishra, G. Prasad, and A. Hossain, "Joint optimization framework for minimization of device energy consumption in transmission rate constrained UAV-assisted IoT network," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9591–9607, Jun. 2022.

[24] R. Karmakar, G. Kaddoum, and S. Chattopadhyay, "Mobility management in 5G and beyond: A novel smart handover with adaptive time-to-trigger and hysteresis margin," *IEEE Trans. Mobile Comput.*, vol. 22, no. 10, pp. 5995–6010, Oct. 2023.

[25] N. N. Ei, P. S. Aung, Z. Han, W. Saad, and C. S. Hong, "Deep-reinforcement-learning-based resource management for task offloading in integrated terrestrial and nonterrestrial networks," *IEEE Internet Things J.*, vol. 12, no. 9, pp. 11977–11993, May2025.
[26] Z. Shao, H. Yang, L. Xiao, W. Su, Y. Chen, and Z. Xiong, "Deep reinforcement learning-based resource management for UAV-assisted mobile edge computing against jamming," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 13358–13374, Dec. 2024.
[27] M. Janner, Q. Li, M. Khanna, and S. Levine, "Offline reinforcement learning as one big sequence modeling problem," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 1273–1286.
[28] Y. He, G. Han, S. Zhu, J. Jiang, Y. Ma, and T. Zhang, "Trust management based on attention-weighted federated deep reinforcement learning for underwater acoustic sensor networks," *IEEE Trans. Mobile Comput.*, vol. 25, no. 6, pp. 8947–8961, Jun. 2026.
[29] A. Martin, I. de-la-Bandera, A. Mendo, J. Outes, J. Ramiro, and R. Barco, "Federated deep reinforcement learning for ENDC optimization," *IEEE Trans. Mobile Comput.*, vol. 24, no. 6, pp. 5525–5535, Jun. 2025.
[30] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.
[31] A. Goldsmith, *Wireless Communications*. Cambridge, U.K.: Cambridge Univ. Press, 2005.
[32] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[33] A. Zhang, R. McAllister, R. Calandra, Y. Gal, and S. Levine, "Learning invariant representations for reinforcement learning without reconstruction," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–17.
[34] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[35] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
[36] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–26.

Zhao Tong (Senior Member, IEEE) received the PhD degree in computer science and technology from the College of Computer Science and Electronic Engineering, Hunan University in China, in 2014. From 2017 to 2018, he was a visiting scholar with Georgia State University. He is currently a full professor with the Hunan Normal University. His main research interests include high-performance computing and super-computing, Big Data management and analytics, resource management, etc. He has published more than 40 research papers in peer-review international conferences and journals, such as *IEEE Transactions on Parallel and Distributed Systems*/SC/*IEEE Transactions on Network and Service Management*/*IEEE Transactions on Industrial Informatics*, etc. He is a distinguished member of the China Computer Federation (CCF).

Shizhen Xiao is currently working toward the master's degree with the College of Information Science and Engineering, Hunan Normal University in China. His primary research interests include center on edge computing, machine learning, and artificial intelligence.

Xi Zhang is currently working toward the MS degree with the College of Information Science and Engineering, Hunan Normal University, Changsha, China. His research interests include resource optimization and task scheduling strategies in mobile edge computing systems.

Lihui Xia is currently working toward the master's degree with the College of Information Science and Engineering, Hunan Normal University, located in China. His research interests include revolve around the areas of mobile edge computing and game theory.

Jing Mei received the PhD degree in computer science from Hunan University, China, in 2015. She is currently an associate professor with the Hunan Normal University, Changsha, China. Her research interests include parallel and distributed computing and cloud computing. She has published 40 research articles in international conference and journals, such as *IEEE Transactions on Computers*/*IEEE Transactions on Parallel and Distributed Systems*/*IEEE Transactions on Services Computing*/*Cluster Computing*/*Journal Grid Computing*/*Journal Supercomput*, etc.

Keqin Li (Fellow, IEEE) received the BS degree in computer science from Tsinghua University, in 1985, and the PhD 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 coauthored more than 1130 journal articles, book chapters and refereed conference papers. 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* (2023-2024) and is among the top 0.02% out of more than 20 million scholars worldwide based on top-cited publications. He is listed in *ScholarGPS Highly Ranked Scholars* (2022-2024) and is among the top 0.002% out of more than 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 2022-2023 *International Science and Technology Cooperation Award* 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).