This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2025.3599885

Distributed Resource Allocation and Coordinated Scheduling for End-Edge-Cloud Collaborative Computing

Changqing Long, Wenchao Meng, *Senior Member, IEEE,* Shizhong Li, Shibo He, *Senior Member, IEEE,* Chaojie Gu, *Member, IEEE,* and Lin Cai, *Fellow, IEEE,*

***Abstract*—Multi-tier computation offloading is crucial to address capacity constraints and improve flexibility for mobile devices. However, existing research on multi-layer computing offloading faces challenges like inefficient resource utilization and poor scalability, particularly in handling diverse computational tasks. To address these challenges, this paper proposes a distributed resource allocation and mixed task offloading framework for end-edge-cloud collaborative systems that support partial and full task offloading modes. First, we propose a three-tier network computing architecture and formulate a task-offloading utility maximization problem by jointly optimizing mixed task-offloading and resource allocation. The proposed problem is a mixed integer nonlinear program (MINLP), which we solve by decomposing it into two subproblems *resource allocation* and *task offloading*. Edge computing resources and bandwidth allocation can be independently optimized at each edge node with a fixed task offloading strategy. Cloud computing resource allocation, while convex, involves a global constraint, which we solve in a decentralized manner using a multi-agent optimization approach. Then, we propose a joint task offloading and resource allocation optimization algorithm, CNO-TORA, to obtain the solution to the formulated problem. The algorithm is supported by strong theoretical guarantees and is almost surely convergent to a globally optimal solution. Experimental results on a real dataset demonstrate that our algorithm is scalable to large-scale networks and outperforms baselines, achieving improvements in average system utility ranging from 4.01%-28.15%.**

***Index Terms*—End-edge-cloud computing; Distributed optimization; Mobile edge computing; Task offloading; Resource allocation.**

I. Introduction

MODERN mobile applications, such as augmented reality, intelligent video processing and AI inference, have imposed increasingly computational workloads on mobile devices (MDs). However, MDs often suffer limited computing capacity and energy constraints, making it hard to meet the latency and other performance requirements of such applications. Mobile Edge Computing (MEC), which enables task offloading from end devices to edge servers (ESs) or cloud servers (CSs), has emerged as a promising solution to bridge this gap. By offloading computational tasks, MDs can effectively alleviate their local processing burden. However, this shift also increases the communication and computational workloads on ESs and CSs, potentially resulting in additional service costs and performance bottlenecks. Consequently, to fully realize the benefits of task offloading while ensuring the sustainability of MDs, it is essential to jointly optimize the offloading strategy, communication, computational resources, and energy consumption across MDs, ESs, and CSs.

In practical scenarios, the task types generated by MDs are highly heterogeneous. Some tasks are indivisible and must be executed as atomic units, while others are divisible and can be split into subtasks [1]. To satisfy diverse computational requirements, MEC systems need to support multiple offloading modes, including partial offloading and full offloading. The integration of mixed offloading modes into MEC networks introduces new challenges. Unlike traditional offloading, mixed offloading requires simultaneously determining the partitioning ratio for divisible tasks and the execution locations for indivisible tasks across the end, edge, and cloud layers. This introduces additional decision variables and complex interdependencies among them. The offloading decision, task splitting, and resource allocation must be jointly optimized. As a result, the problem becomes a high-dimensional mixed-integer nonlinear programming (MINLP) formulation with strong coupling among variables, significantly increasing the difficulty of designing efficient solution methods.

Although significant efforts have been dedicated to developing task offloading and resource allocation strategies to enhance the performance of end–edge–cloud networks [19–22], these studies support only a single offloading mode. i.e., either full offloading or partial offloading. Moreover, they typically rely on centralized optimization methods to solve the formulated problems, which incur high communication and computation overheads, thereby limiting their scalability in large-scale edge computing networks. These limitations highlight the necessity of an end–edge–cloud network that supports mixed offloading modes, along with a distributed optimization framework for jointly optimizing offloading decisions and resource allocation. The main contributions of this work are listed as follows.

1) This paper proposes an end-edge-cloud collaborative computing network that supports mixed offloading modes, including both partial and full offloading, and formulates an optimization problem to maximize users' offloading benefits. Compared to existing architectures that support either full or partial offloading, our proposed network is more general and flexible, enabling the execution of a wider

C. Long, W. Meng, S. Li, S. He and C. Gu are with the State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China.

L. Cai is with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC V8P 5C2, Canada.

TABLE I
COMPARISON OF THE PROPOSED METHOD AND PREVIOUS WORKS

		Ours	Previous Works															
			[2]	[3]	[4, 5]	[6]	[7]	[8]	[9]	[1]	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]
Architecture	End-Edge-Cloud Collaboration	✓			✓									✓			✓	✓	✓
	Multi-Edge Servers	✓	✓	✓		✓	✓			✓		✓	✓	✓	✓	✓	✓	✓	✓
	Mixed Offloading	✓					✓	✓	✓	✓	✓	✓	✓						
	Resource Allocation	✓					✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
Objective	Energy Consumption	✓	✓		✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
	Latency	✓	✓	✓	✓	✓	✓	✓		✓			✓	✓	✓	✓	✓	✓	✓
	Energy-Delay Trade-off	✓	✓		✓	✓									✓		✓		
Approach	Heuristic													✓					
	Deep Learning			✓	✓			✓					✓					✓	
	Distributed Optimization	✓	✓			✓									✓	✓	✓	✓	✓
	Other Optimization						✓		✓	✓	✓	✓							

range of task types.

2) To address the complexity of the formulated problem and ensure scalability in large-scale networks, we decompose the problem into subproblems and solve them in a distributed manner. Most notably, we design a multi-agent optimization approach to solve the cloud computing resource allocation problem under global constraints, and propose an intelligent algorithm, CNO-TORA, to jointly optimize task offloading and resource allocation.
3) Theoretical analyses, including scalability, convergence, and computational complexity, as well as experimental evaluations based on real-world datasets, are provided to validate the effectiveness of our approach.

The remainder of this paper is organized as follows. The related work is reviewed in Section II. The network model is presented in Section III, and the problem formulation is given in Section IV. Section V provides the distributed solution to the formulated optimization problem. Evaluation results are provided in Section VI. In the end, the conclusions are given in Section VII.

II. RELATED WORK

In this section, we review the related work from three aspects: task offloading, mixed offloading, and joint offloading and resource allocation in MEC networks. The main distinctions between our proposed method and existing studies are summarized in Table I.

A. Task Offloading in MEC

Task offloading is a core mechanism in MEC and is commonly classified into full offloading and partial offloading. Many studies have focused on these two strategies to optimize the performance of MEC systems in terms of latency and energy consumption. For example, based on game-theoretic methods [2], reinforcement learning [3–5], or distributed optimization [6], a variety of full offloading strategies have been developed for different network scenarios, such as UAV-assisted MEC, satellite-edge networks, and cooperative edge computing networks. And for partial offloading, Diamanti *et al*. [23] proposed a partial offloading strategy for RSMA-enabled MEC systems, where each task can be arbitrarily partitioned and processed across multiple ESs. Based on delay-optimality conditions, Ren *et al*. derived closed-form expressions for the optimal offloading ratio between local and edge computing in [24], and later extended this to the edge–cloud collaboration scenario in [22]. Subsequently, Zhu *et al*. [21] and Liu *et al*. [19] derived closed-form solutions for optimal task partitioning ratio in the satellite-terrestrial networks and end–edge–cloud collaborative networks, respectively. Furthermore, partial offloading strategies have also been investigated in vehicular edge computing networks [25], UAV-assisted MEC networks [26], and heterogeneous edge networks [27].

It is worth noting that all the above studies focus on either full or partial offloading in isolation. The joint design of mixed offloading strategies, which enables the flexible coexistence and coordination of both modes, has not yet been addressed.

B. Mixed task offloading in MEC

Current research on mixed offloading in MEC networks generally follows two main directions. The first focuses on modeling and optimizing full and partial offloading separately within the same MEC network [7–9]. The second enables the coexistence of both modes, allowing tasks to be flexibly offloaded as full or partial depending on task types [1, 10–12].

For the first approach, Feng *et al.* [7] and Chen *et al.* [8] optimized task latency in vehicular and wireless-powered MEC networks, respectively, by separately designing strategies for full and partial offloading. Chen *et al.* [9] focused on multi-user MEC systems and aimed to minimize energy consumption under different offloading strategies. For the second approach, Chen *et al.* [1] considered user satisfaction and the coexistence of mixed tasks in fog computing systems. Wu *et al.* [10] formulated a residual energy maximization problem for a wireless powered MEC system with mixed offloading. Li *et al.* [11] investigated mixed offloading in an edge–cloud collaborative network to reduce overall system energy consumption. Chen *et al.* [12] studied a space–air-ground network and proposed a mixed offloading strategy to jointly minimize delay and energy consumption.

Our work focuses on the coexistence of full and partial offloading. Unlike previous studies [1, 10–12], in which [1, 10, 12] do not consider the cloud tier, whereas[11] ignores

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2025.3599885

the computing capabilities of MDs. Thus, how to fully exploit the complementary strengths of MDs, ESs, and CSs within a unified mixed-offloading framework remains an open issue.

C. Joint offloading and resource allocation in MEC

Task offloading and resource allocation are inherently coupled in MEC networks, as the effectiveness of offloading decisions largely depends on the allocation of computational and communication resources. Consequently, the joint optimization of these two aspects is of vital importance and has become a key focus of extensive research effort, resulting in a wide variety of optimization methods.

Classical approaches such as quadratically constrained quadratic programming (QCQP) [28], branch-and-bound techniques [29], semi-definite programming (SDP) [30], and Lyapunov optimization [31] have been widely adopted to derive joint task offloading and resource allocation schemes. While these methods can provide near-optimal or optimal solutions, the computational burden escalates rapidly with network size. To reduce complexity, Bi et al. [13, 32, 33] designed a series of fast and low-complexity heuristic algorithms tailored to optimize offloading and resource allocations across diverse MEC scenarios, effectively reducing energy consumption and overall system cost. Complementing these model-driven techniques, learning-based methods, particularly deep reinforcement learning (DRL), have also received growing attention. Chen et al. [34] introduced a digital-twin-assisted DRL framework for efficient task offloading and bandwidth allocation. Liu et al. [35] proposed a DRL-based dual timescale scheme that jointly optimizes service caching, offloading, communication, and computing resources. Heuristic and DRL-based algorithms generally lack rigorous theoretical guarantees and often provide feasible rather than optimal solutions. Notably, the above solutions are centralized, which limits their scalability in large-scale edge computing networks.

Several studies have proposed distributed optimization solutions to support scalability in MEC networks. For example, Fan et al. [18] proposed a two-stage distributed latency minimization scheme for full offloading and resource allocation in end–edge–cloud network. Kim et al. [14] further introduced a pricing-based distributed latency minimization solution for full offloading and resource allocation in MEC systems. The authors of [6, 15] developed ADMM-based distributed algorithms to jointly optimize task offloading and resource allocation. In addition, game-theoretic approaches and multi-agent DRL techniques have also been employed to design distributed solutions [16, 17, 36].

However, the above distributed solutions are primarily designed for the joint optimization of either partial offloading and resource allocation or full offloading and resource allocation. They cannot be directly applied to the joint optimization of mixed offloading and resource allocation in end–edge–cloud networks. Moreover, ADMM-based methods [6, 15] require the exchange of local variables among neighboring nodes, which may disclose sensitive information. This paper proposes a distributed method that only exchanges Lagrange multipliers only, thereby providing basic data privacy protection.

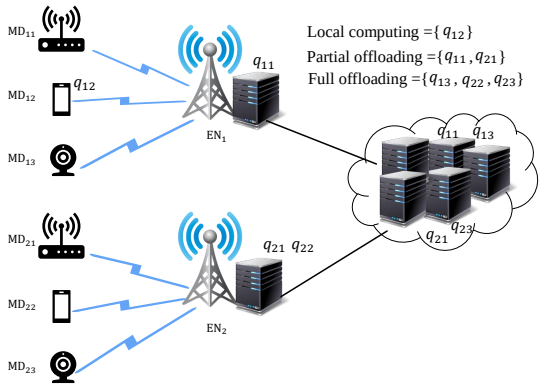


Fig. 1. An example of mixed-task coexistence is shown in an MEC network with $\mathcal{M} = 6$ MDs, $\mathcal{N} = 2$ ENs, and a CS. MD_{11} and MD_{21} have divisible tasks (q_{11}, q_{12}) that are split and executed across the local device, edge, and cloud. The remaining tasks are indivisible and must be fully processed at the local device, an EN, or the cloud.

III. Network Model

We consider an end-edge-cloud computing network consisting of one CS, n base stations (BSs), denoted by the set $\mathcal{N} \triangleq \{1, 2, \cdots, n\}$, and $\mathcal{M}$ MDs, represented as $\mathcal{M} \triangleq \{\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_n\}$. Each subset $\mathcal{M}_n \triangleq \{1, 2, \cdots, m_n\}$ denotes the set of MDs associated with BS n. In this network, each BS is equipped with a MEC server (MECS) to provide computing services for the MDs with resource-constrained, such as tablets and smartphones. An edge node (EN) is composed of one BS and one co-located MECS. Each EN has a group of MDs, $\mathcal{M}_n$, connected via wireless cellular links, and each MD has a computing task that needs to be computed. To support diverse computational requirements, this network considers the coexistence of divisible and indivisible tasks.

The task of $\mathrm{MD}_{n,m}$, severed by EN n, is modeled as $q_{n,m} \triangleq \{\mu_{n,m}, \gamma_{n,m}, \kappa_{n,m}\}$, where $\mu_{n,m}$ (in bits) is the input data-size and $\gamma_{n,m}$ (in cycle/bit) represents the number of CPU cycles required to compute this task's one-bit data. $\kappa_{n,m}$ represents the task type. $\kappa_{n,m} = 1$ indicates that the task is divisible, $\kappa_{n,m} = 0$, otherwise. Fig. 1 provides an illustrative example of mixed-task coexistence in a MEC network consisting of $\mathcal{M} = 6$ MDs, $\mathcal{N} = 2$ ENs, and a CS.

A. Communication Model

In the proposed network, task transmission involves two communication stages: from the MDs to the ENs and from the ENs to the CS. The communication between EN n and the CS utilizes the wired backhaul link. Similarly to [22], let $\Re_n^{cs}$ (in Mbits/sec) be the backhaul communication capacity. The orthogonal frequency division multiple access (OFDMA) scheme is employed to avoid interference between signals from different MDs and ENs. For simplicity, we adopt the assumption from [20] that all MDs transmit task data at maximum power, and the channel between MDs and ENs is quasi-static during task transmission but may change in subsequent time slots. In addition, the return latency is ignored as the final results are much smaller than the original data.

Using Shannon's formula, the achievable uplink data rate for $\text{MD}_{n,m}$ served by EN n is given by

$$w_{n,m}\Re^{es}_{n,m}, \tag{1}$$

where w_{nm} is the proportion of the bandwidth allocated from EN n to $\text{MD}_{n,m}$, and

$$\Re^{es}_{n,m} = B^{es}_n \log_2\left(1+\frac{P_{n,m}h_{n,m}}{\sigma_n}\right), \tag{2}$$

B^{es}_n (in Hz) is the available bandwidth at EN n, $P_{n,m}$ is the transmission power of $\text{MD}_{n,m}$, and $h_{n,m}$ is the channel gain between $\text{MD}_{n,m}$ and EN n. σ_n is the background noise power.

TABLE II
NOTATIONS USED IN THIS PAPER

Notation	Definition
$\mathcal{M}$	Set of all MDs
$\mathcal{N}$	Set of ENs
$\mathcal{M}_n$	Set of MDs associated with EN n,
$\mathcal{M}^0_n$	Set of MDs with partial offloading associated with EN n
$\mathcal{M}^1_n$	Set of MDs with full offloading associated with EN n
$\mu_{n,m}, \gamma_{n,m}, \kappa_{n,m}$	Input data size, CPU cycles per bit, and task type
$\alpha_{n,m}, \zeta_{n,m}, \theta_{n,m}$	Task splitting ratio for local, edge, and cloud
$I^{x,l}_{n,m}, I^{x,e}_{n,m}, I^{x,c}_{n,m}$	Indicator functions for local, edge, and cloud execution
$\delta^1_{n,m}, \delta^2_{n,m}$	Latency and energy consumption trade-off weights
$\ell_{n,m}$	Utility weights
$x_{n,m}$	Decision variable for full offloading
B^{es}_n	Bandwidth at EN n
$w_{n,m}$	Bandwidth allocation ratio between EN n MD m
$y_{n,m}$	Computing resource allocation ratio between EN n MD m
$z_{n,m}$	Computing resource allocation ratio between CS and MD m
$T^{loc}_{n,m}$	Delay for $\text{MD}_{n,m}$ execution its entire task locally
$T^{e,trans}_{n,m}$	Delay for $\text{MD}_{n,m}$ transferring its entire task data to EN n
$T^{e,comp}_{n,m}$	Delay for $\text{MD}_{n,m}$ executing its entire task at EN n
$T^{c,trans}_{n,m}$	Delay for $\text{MD}_{n,m}$ transferring its entire task data to CS
$T^{c,comp}_{n,m}$	Delay for $\text{MD}_{n,m}$ executing its entire task in the cloud
$E^{loc}_{n,m}$	Energy consumption for $\text{MD}_{n,m}$ executing its entire task locally

B. Computing Model

Let $f^{loc}_{n,m}$ (in cycle/s), f^{es}_n (in cycle/s) and f^{cs} (in cycle/s) represent the computing capacity of $\text{MD}_{n,m}$ EN n and CS, respectively. $\mathcal{M}^0_n \triangleq \{1,2,\cdots,k_n\}$ and $\mathcal{M}^1_n \triangleq \{k_n+1, k_n+2,\cdots,m_n\}$ denote the sets of MDs performing partial and full offloading, respectively. For dividable tasks, $\alpha_{n,m}$, $\zeta_{n,m}$ and $\theta_{n,m}(n\in\mathcal{N}, m\in\mathcal{M}^0_n)$ are the proportion of data processed at the $\text{MD}_{n,m}$, EN n, and CS, respectively. And for fully offloaded tasks, $x_{n,m}=0$, $x_{n,m}=1/2$ and $x_{n,m}=1(n\in\mathcal{N}, m\in\mathcal{M}^1_n)$ indicated that the task is processed at $\text{MD}_{n,m}$, EN n, and CS, respectively. Next, we model the transmission and computation delays incurred by task offloading in local, edge, and cloud computing.

1) Local computing: For task $q_{n,m}(n\in\mathcal{N}, m\in\mathcal{M}_n)$, the MD utilizes its full computation capacity for local processing. Thus, the delay of local computing can be estimated as

$$t^{loc}_{n,m} = \left(\kappa_{n,m}, 1-\kappa_{n,m}\right)\begin{pmatrix}\alpha_{n,m}\\ I^{x,l}_{n,m}\end{pmatrix} T^{loc}_{n,m}, \tag{3}$$

where $T^{loc}_{n,m} = \mu_{n,m}\gamma_{n,m}/f^{loc}_{n,m}$, $I^{x,l}_{n,m} = (1-2x_{n,m})(1-x_{n,m})$.

2) Edge computing: Whether partially or fully offloaded, the data to be processed in ENs or the CS must first be transmitted to the ENs via the wireless channel. Since MDs are close to EN, the propagation delay can be neglected. Then, combining (2), the transmission delay for task $q_{n,m}$ can be modeled as

$$t^{e,trans}_{n,m} = \left(\kappa_{n,m}, 1-\kappa_{n,m}\right)\begin{pmatrix}1-\alpha_{n,m}\\ I^{x,e}_{n,m}\end{pmatrix} T^{e,trans}_{n,m}, \tag{4}$$

where $T^{e,trans}_{n,m} = \mu_{n,m}/w_{n,m}\Re^{es}_{n,m}$, $I^{x,e}_{n,m} = 4x_{n,m}(1-x_{n,m})$. Let $y_{n,m}$ denote the fraction of computation resources allocated by EN n to $\text{MD}_{n,m}$. Accordingly, the partial or full data of task $q_{n,m}$ generated by $\text{MD}_{n,m}$ is processed using $y_{n,m}f^{es}_n$ computing resources in EN n. So, the computing delay can be expressed as

$$t^{e,comp}_{n,m} = \left(\kappa_{n,m}, 1-\kappa_{n,m}\right)\begin{pmatrix}\zeta_{n,m}\\ I^{x,e}_{n,m}\end{pmatrix} T^{e,comp}_{n,m}, \tag{5}$$

in which $T^{e,comp}_{n,m} = \frac{\mu_{n,m}\gamma_{n,m}}{y_{n,m}f^{es}_n}$. To sum up, the delay experienced by $\text{MD}_{n,m}$ for data processed on EN n is

$$t^{e}_{n,m} = t^{e,trans}_{n,m} + t^{e,comp}_{n,m}. \tag{6}$$

3) Cloud computing: For the remaining data $\theta_{n,m}\mu_{n,m}$ of divisible tasks or the full date of indivisible tasks, if the offloaded data is chosen to be processed at the CS, this data needs to be transferred from the EN to the CS, and then the transmission delay can be modeled as

$$t^{c,trans}_{n,m} = \left(\kappa_{n,m}, 1-\kappa_{n,m}\right)\begin{pmatrix}\theta_{n,m}\\ I^{x,c}_{n,m}\end{pmatrix} T^{c,trans}_{n,m}, \tag{7}$$

where $T^{c,trans}_{n,m} = \mu_{n,m}/\Re^{cs}_n$, $I^{x,c}_{n,m} = x_{n,m}(2x_{n,m}-1)$. Due to the considerable distance between the CS and ENs, the propagation delay of links between ENs and CS cannot be overlooked. Considering both uplink and downlink communication, the round-trip propagation delay for EN n is

$$t^{c,trip}_{n,m} = \left(\kappa_{n,m}, 1-\kappa_{n,m}\right)\begin{pmatrix}1\\ I^{x,c}_{n,m}\end{pmatrix}\frac{2D_n}{C}, \tag{8}$$

in which D_n represents the distance between the CS and the EN n, and C denotes the speed of light. After receiving the data from ENs, the CS will allocate the available computation resource to $\text{MD}_{n,m}$ for processing its tasks. Let $z_{n,m}$ represent the fraction of computation resources allocated by CS n to $\text{MD}_{n,m}$. The partial or full data of task $q_{n,m}$ generated by $\text{MD}_{n,m}$ is processed using $z_{n,m}f^{cs}$ computing resources, and the computing delay is given by

$$t^{c,comp}_{n,m} = \left(\kappa_{n,m}, 1-\kappa_{n,m}\right)\begin{pmatrix}\theta_{n,m}\\ I^{x,c}_{n,m}\end{pmatrix} T^{c,comp}_{n,m}, \tag{9}$$

where $T^{c,comp}_{n,m} = \frac{\mu_{n,m}\gamma_{n,m}}{z_{n,m}f^{cs}}$. Similarly, the delay experienced by $\text{MD}_{n,m}$ for data processed on CS is

$$\begin{aligned} t^{c}_{n,m} = &\left(\kappa_{n,m}, 1-\kappa_{n,m}\right)\begin{pmatrix}1-\alpha_{n,m}\\ I^{x,c}_{n,m}\end{pmatrix} T^{e,trans}_{n,m}\\ &+ t^{c,trans}_{n,m} + t^{c,comp}_{n,m} + t^{c,trip}_{n,m}. \end{aligned} \tag{10}$$

For divisible tasks $q_{n,m}$ generated by $\text{MD}_{n,m}$, local, edge, and cloud computing are processed in parallel. Consequently, the total latency experienced by $\text{MD}_{n,m}$ from task offloading and processing across local, EN, and CS can be expressed as

$$t^{off}_{n,m} = \begin{cases} \max\left\{t^{l,p}_{n,m}, t^{e,p}_{n,m}, t^{c,p}_{n,m}\right\}, \kappa_{n,m}=1\\ t^{l,b}_{n,m}+t^{e,b}_{n,m}+t^{c,b}_{n,m}, \kappa_{n,m}=0, \end{cases} \tag{11}$$

where $t^{l,p}_{n,m} = \alpha_{n,m} T^{loc}_{n,m}$, $t^{e,p}_{n,m} = (1-\alpha_{n,m}) T^{e,trans}_{n,m} + \zeta_{n,m} T^{e,comp}_{n,m}$, $t^{c,p}_{n,m} = (1-\alpha_{n,m}) T^{e,trans}_{n,m} + \theta_{n,m}(T^{c,trans}_{n,m} + T^{c,comp}_{n,m}) + \frac{2D_n}{C}$; $t^{l,b}_{n,m} = I^{x,l}_{n,m} T^{loc}_{n,m}$, $t^{e,b}_{n,m} = I^{x,e}_{n,m}(T^{e,trans}_{n,m} + T^{e,comp}_{n,m})$, $t^{c,b}_{n,m} = I^{x,c}_{n,m}\left(T^{e,trans}_{n,m} + T^{c,trans}_{n,m} + T^{c,comp}_{n,m} + \frac{2D_n}{C}\right)$.

C. Energy Consumption Model

In the end-edge-cloud computing network, the energy consumption of MD is determined by the computation and transmission processes.

1) Local Computation Energy: The energy consumed by the MD during local computation depends on the CPU frequency and the number of cycles required to process the task. For a user associated with EN n, the local computing energy consumption can be modeled as

$$\widehat{E}_{n,m} = \left(\kappa_{n,m}, 1-\kappa_{n,m}\right) \begin{pmatrix} \alpha_{n,m} \\ I^{x,l}_{n,m} \end{pmatrix} E^{loc}_{n,m}, \tag{12}$$

where $E^{loc}_{n,m} = k \mu_{n,m} \gamma_{n,m} (f^{loc}_{n,m})^2$, and k is a hardware-dependent energy coefficient.

2) Transmission Energy: The energy consumed during task transmission depends on the data size to be offloaded and the transmission power of the MD. The energy consumption model for transmitting the offloaded task is given by

$$\overline{E}_{n,m} = \left(\kappa_{n,m}, 1-\kappa_{n,m}\right) \begin{pmatrix} 1-\alpha_{n,m} \\ 1-I^{x,l}_{n,m} \end{pmatrix} P_{n,m} T^{e,trans}_{n,m}, \tag{13}$$

The total energy consumption of $\text{MD}_{n,m}$, considering both computation and transmission energy, is given by

$$E_{n,m} = \widehat{E}_{n,m} + \overline{E}_{n,m}. \tag{14}$$

D. Utility Function Definition

The user offloading utility quantifies the benefit of task offloading by considering both latency reduction and energy efficiency. The utility function for $\text{MD}_{n,m}$ associated with EN n is defined as

$$U_{n,m} = \delta^1_{n,m} \frac{T^{loc}_{n,m} - t^{off}_{n,m}}{T^{loc}_{n,m}} + \delta^2_{n,m} \frac{E^{loc}_{n,m} - E_{n,m}}{E^{loc}_{n,m}}, \tag{15}$$

where $\delta^1_{n,m}$ and $\delta^2_{n,m} (0 \le \delta^1_{n,m}, \delta^2_{n,m}$ and $\delta^1_{n,m} + \delta^2_{n,m} = 1)$ are weighting coefficients balancing the importance of latency and energy efficiency; $T^{loc}_{n,m}$ is the delay for local processing defined in (3), and $t^{off}_{n,m}$ is given in (11); $E^{loc}_{n,m}$ and $E_{n,m}$ are given in (12) and (14), respectively.

For ease of reference, Table II provides a summary of the key notations used in the paper.

IV. Problem Formulation

In this paper, the goal is to maximize the total utility for all users in the network. We formulate the optimization problem as follows.

$$\mathcal{P}_1: \max_{\boldsymbol{\alpha},\boldsymbol{\zeta},\boldsymbol{\theta},\boldsymbol{X},\boldsymbol{W},\boldsymbol{Y},\boldsymbol{Z}} \sum_{n\in\mathcal{N}} \sum_{m\in\mathcal{M}_n} \ell_{n,m} U_{n,m} \tag{16a}$$

$$\text{s.t.} \quad \alpha_{n,i}, \zeta_{n,i}, \theta_{n,i} \in (0,1), \quad \forall n \in \mathcal{N}, \forall i \in \mathcal{M}^0_n, \tag{16b}$$

$$\alpha_{n,i} + \zeta_{n,i} + \theta_{n,i} = 1, \quad \forall n \in \mathcal{N}, \forall i \in \mathcal{M}^0_n, \tag{16c}$$

$$t^{l,p}_{n,m} = t^{e,p}_{n,m} = t^{c,p}_{n,m}, \quad \forall n \in \mathcal{N}, \forall i \in \mathcal{M}^0_n, \tag{16d}$$

$$x_{n,j} \in \{0, 1/2, 1\}, \quad \forall n \in \mathcal{N}, \forall j \in \mathcal{M}^1_n, \tag{16e}$$

$$\sum_{m\in\mathcal{M}_n} w_{n,m} \le 1, \quad w_{n,m} \ge 0, \ \forall n \in \mathcal{N}, \tag{16f}$$

$$\sum_{m\in\mathcal{M}_n} y_{n,m} \le 1, \quad y_{n,m} \ge 0, \ \forall n \in \mathcal{N}, \tag{16g}$$

$$\sum_{n\in\mathcal{N}} \sum_{m\in\mathcal{M}_n} z_{n,m} \le 1, \quad z_{n,m} \ge 0, \tag{16h}$$

where the parameter vectors $\boldsymbol{\alpha}$, $\boldsymbol{\zeta}$, $\boldsymbol{\theta}$, $\boldsymbol{X}$, $\boldsymbol{W}$, $\boldsymbol{Y}$ and $\boldsymbol{Z}$ are defined by $\boldsymbol{\alpha} = \{\alpha_{n,i}\}$, $\boldsymbol{\zeta} = \{\zeta_{n,i}\}$, $\boldsymbol{\theta} = \{\theta_{n,i}\}$, $\boldsymbol{X} = \{x_{n,j}\}$, $\boldsymbol{W} = \{w_{n,j}\}$, $\boldsymbol{Y} = \{y_{n,m}\}$ and $\boldsymbol{Z} = \{z_{n,m}\}$, respectively. $\boldsymbol{\alpha}, \boldsymbol{\zeta}, \boldsymbol{\theta} \in \mathcal{R}^{\sum_{n\in\mathcal{N}} \mathcal{M}^0_n}$, $\boldsymbol{X} \in \mathcal{R}^{\sum_{n\in\mathcal{N}} \mathcal{M}^1_n}$ and $\boldsymbol{W}, \boldsymbol{Y}, \boldsymbol{Z} \in \mathcal{R}^{\mathcal{M}} (\mathcal{M} = \sum_{n\in\mathcal{N}} \mathcal{M}^0_n + \sum_{n\in\mathcal{N}} \mathcal{M}^1_n)$. (16b) and (16c) are the task splitting ratios constraint. (16d) ensures that the divisible tasks generated by MDs can achieve minimal latency. This condition has been proved in [19] and [21]. Constrain (16e) is an indicator variable. The constraint (16f) guarantees that the total bandwidth allocated to the associated users with EN n does not exceed its available bandwidth capacity. Constrains (16g) and (16h) ensure that the computation resources allocated to MDs do not exceed the maximum available resources of each EN and CS, respectively. $\ell_{n,m}$ is a utility weight that indicates the priority of the task. The problem $\mathcal{P}_1$ is a MINLP problem, as it involves a nonlinear and nonconvex objective function with discrete variables $\boldsymbol{X}$ and continuous variables $\boldsymbol{\alpha}$, $\boldsymbol{\zeta}$, $\boldsymbol{\theta}$, $\boldsymbol{W}$, $\boldsymbol{Y}$, $\boldsymbol{Z}$. Solving such problems typically requires exponential time complexity to find the optimal solution.

V. Distributed Resource Allocation and Mixed Task Offloading Scheme

In this section, we propose a distributed resource allocation and mixed task offloading framework for end-edge-cloud collaborative systems. We first transform the original discontinuous optimization problem into a continuous optimization problem based on latency-optimality conditions. Then, we decompose the problem into two subproblems: *resource allocation optimization* and *task offloading optimization*. For *resource allocation optimization*, edge computing resources and bandwidth allocation can be independently optimized at each EN with a fixed task offloading strategy, allowing closed-form solutions to be efficiently obtained. Cloud computing resource allocation is a convex optimization, but it involves a global constraint. To solve it in a decentralized manner, we employ a multi-agent optimization algorithm based on projected neural network (PNN). For *task offloading optimization*, we leverage latency-optimality conditions to derive closed-form solutions to both local and edge task offloading ratios. By substituting these solutions into the original problem and introducing relaxation variables, we reformulate the problem into a more tractable form. We then design the collaborative neurodynamic optimization (CNO) algorithm to solve the problem based on the penalty method. Furthermore, we propose CNO-TORA to jointly optimize mixed task offloading strategies and resource allocation. Finally, we provide scalability analysis, conver-

gence analysis, and complexity analysis for the proposed algorithm. The algorithmic schematic is presented in Fig. 2.

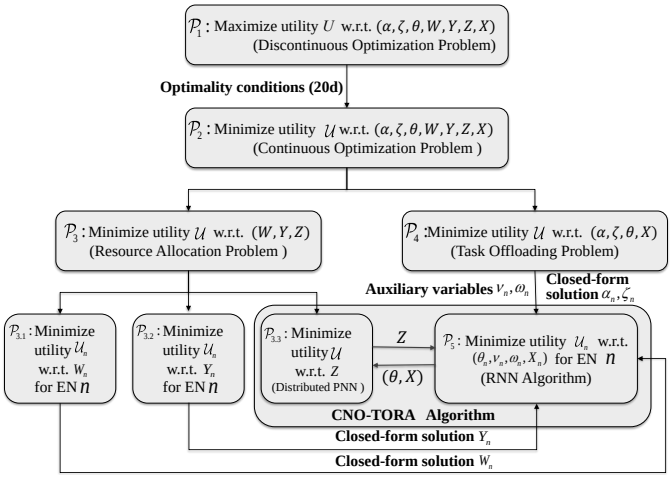


Fig. 2. Schematic of the algorithms for optimizing task offloading benefits in an end-edge-cloud collaborative system.

A. Problem transformation

Combined with (11), (14) and (16d), the objective function of EN n is reformuted as follows.

$$U_n = \sum_{m\in\mathcal{M}_n}\left(\phi^1_{n,m}T^{loc}_{n,m} + \phi^2_{n,m}E^{loc}_{n,m}\right) - \mathcal{U}_n \tag{17}$$

where $\phi^1_{n,m} = \ell_{n,m}\delta^1_{n,m}/T^{loc}_{n,m}$, $\phi^2_{n,m} = \ell_{n,m}\delta^2_{n,m}/E^{loc}_{n,m}$, and

$$\begin{aligned}\mathcal{U}_n = \sum_{m\in\mathcal{M}_n}\Big[&g^{(\alpha,x)}_{n,m}T^{e,trans}_{n,m} + g^{(\zeta,x)}_{n,m}T^{e,comp}_{n,m} \\ &+ g^{(\theta,x)}_{n,m}T^{c,comp}_{n,m}\Big] + \sum_{m\in\mathcal{M}^0_n} f^{(\alpha,\theta)}_{n,m} + \sum_{m\in\mathcal{M}^1_n} f^{(x)}_{n,m},\end{aligned} \tag{18}$$

in which $g^{(\alpha,x)}_{n,m} = \left(\kappa_{n,m}, 1-\kappa_{n,m}\right)\begin{pmatrix}1-\alpha_{n,m}\\ 1-I^{x,l}_{n,m}\end{pmatrix}(\phi^1_{n,m} + \phi^2_{n,m}P_{n,m})$, $g^{(\zeta,x)}_{n,m} = \left(\kappa_{n,m}, 1-\kappa_{n,m}\right)\begin{pmatrix}0.5\zeta_{n,m}\\ I^{x,e}_{n,m}\end{pmatrix}\phi^1_{n,m}$, $g^{(\theta,x)}_{n,m} = \left(\kappa_{n,m}, 1-\kappa_{n,m}\right)\begin{pmatrix}0.5\theta_{n,m}\\ I^{x,c}_{n,m}\end{pmatrix}\phi^1_{n,m}$, $f^{(\alpha,\theta)}_{n,m} = \alpha_{n,m}\phi^2_{n,m}E^{loc}_{n,m} + \frac{1}{2}\theta_{n,m}\phi^1_{n,m}T^{c,trans}_{n,m} + \phi^1_{n,m}\frac{D_n}{C}$, and $f^{(x)}_{n,m} = I^{x,l}_{n,m}(\phi^1_{n,m}T^{loc}_{n,m} + \phi^2_{n,m}E^{loc}_{n,m}) + I^{x,c}_{n,m}\left(T^{c,trans}_{n,m} + \frac{2D_n}{C}\right)$.

It should be pointed out that the constraint (16e) is equivalently transformed into $h_{n,m}(x_{n,m}) \triangleq x_{n,m}(1/2 - x_{n,m})(1 - x_{n,m}) = 0$. It follows from (17), problem (16) is rewritten as follows.

$$\mathcal{P}_2: \quad \min_{\boldsymbol{\alpha},\boldsymbol{\zeta},\boldsymbol{\theta},\boldsymbol{X},\boldsymbol{W},\boldsymbol{Y},\boldsymbol{Z}} \sum_{n\in\mathcal{N}}\mathcal{U}_n \tag{19a}$$

$$s.t. \quad (16b), (16c), (16d), (16f), (16g), (16h), \tag{19b}$$

$$h_{n,m}(x_{n,m}) = 0, \ \forall n\in\mathcal{N}, \forall m\in\mathcal{M}^1_n. \tag{19c}$$

B. Problem decomposition

The problem $\mathcal{P}_2$ remains challenging to solve due to the coupling between variables. To address this, we decompose the problem $\mathcal{P}_2$ into two subproblem, $\mathcal{P}_3$ and $\mathcal{P}_4$, as follows.

$$\mathcal{P}_3: \quad \min_{\boldsymbol{W},\boldsymbol{Y},\boldsymbol{Z}} \mathcal{U}(\boldsymbol{W},\boldsymbol{Y},\boldsymbol{Z}) := \sum_{n\in\mathcal{N}}\mathcal{U}_n \tag{20a}$$

$$s.t. \quad (16f), (16g), (16h), \tag{20b}$$

and

$$\mathcal{P}_4: \quad \min_{\boldsymbol{\alpha},\boldsymbol{\zeta},\boldsymbol{\theta},\boldsymbol{X}} \mathcal{U}(\boldsymbol{\alpha},\boldsymbol{\zeta},\boldsymbol{\theta},\boldsymbol{X}) := \sum_{n\in\mathcal{N}}\mathcal{U}_n \tag{21a}$$

$$s.t. \quad (16b), (16c), (16d), \tag{21b}$$

$$\boldsymbol{h}_n(\boldsymbol{X}_n) = \boldsymbol{0}_{\mathcal{M}^1_n}, \ \forall n\in\mathcal{N}, \tag{21c}$$

where $\boldsymbol{h}_n(\boldsymbol{X}_n) = \boldsymbol{X}_n \circ (\boldsymbol{e} - \boldsymbol{X}_n) \circ (0.5\boldsymbol{e} - \boldsymbol{X}_n)$, $\boldsymbol{e}$ is a vector of ones, $\circ$ stands for Hadamard product.

Subproblem $\mathcal{P}_3$ focuses on optimizing $\boldsymbol{W}$, $\boldsymbol{Y}$ and $\boldsymbol{Z}$, given a fixed offloading strategy $\{\boldsymbol{\alpha}^*, \boldsymbol{\zeta}^*, \boldsymbol{\theta}^*, \boldsymbol{X}^*\}$. The subproblem $\mathcal{P}_4$ is then formulated to optimize the task offloading variable $\{\boldsymbol{\alpha}, \boldsymbol{\zeta}, \boldsymbol{\theta}, \boldsymbol{X}\}$, given the fixed value of $\boldsymbol{W}^*$, $\boldsymbol{Y}^*$ and $\boldsymbol{Z}^*$. The decomposition approach provides a practical and efficient solution framework for optimizing resource allocation and task offloading in end-edge-cloud systems. The following sections detail the solution methods for $\mathcal{P}_3$ and $\mathcal{P}_4$.

C. Distribute Resources allocated scheme

This section focuses on solving the subproblem $\mathcal{P}_3$ using a distributed approach with given offloading strategy $\{\boldsymbol{\alpha}^*, \boldsymbol{\zeta}^*, \boldsymbol{\theta}^*, \boldsymbol{X}^*\}$.

1) Bandwidth Resource Allocation with Fixed Task Offloading Strategy: In problem (20), we ignore the terms that are unrelated to the optimization variables. Consequently, the subproblem associated with optimizing $\boldsymbol{W}$ is reformulated as follows.

$$\mathcal{P}_{3.1}: \quad \min_{\boldsymbol{W}} \sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}_n} g^{(\alpha,x)}_{n,m}\frac{\mu_{n,m}}{w_{n,m}\Re^{es}_{n,m}} \tag{22a}$$

$$s.t. \quad \sum_{m\in\mathcal{M}_n} w_{n,m} \le 1, \ \forall n\in\mathcal{N} \tag{22b}$$

$$w_{n,m} \ge 0. \tag{22c}$$

Since bandwidth allocation at each EN is independent of other ENs, we can decompose the global problem (22) into independent subproblems for each individual EN n, formulated as follows.

$$\mathcal{P}_{3.1.1}: \quad \min_{\boldsymbol{W}_n} \sum_{m\in\mathcal{M}_n} g^{(\alpha,x)}_{n,m}\frac{\mu_{n,m}}{w_{n,m}\Re^{es}_{n,m}} \tag{23a}$$

$$s.t. \quad (22b), (22c). \tag{23b}$$

We employ the KKT conditions to derive the optimal solution for problem $\mathcal{P}_{3.1.1}$. The detailed solution process is omitted here, as it can be computed straightforwardly. The closed-form expression for the bandwidth allocation ratio is derived by

$$w_{n,m} = \frac{\sqrt{g^{(\alpha,x)}_{n,m}\mu_{n,m}/\Re^{es}_{n,m}}}{\sum_{m\in\mathcal{M}_n}\sqrt{g^{(\alpha,x)}_{n,m}\mu_{n,m}/\Re^{es}_{n,m}}}. \tag{24}$$

2) Edge Computing Resource Allocation with Fixed Task Offloading Strategy: Similar to the problem $\mathcal{P}_{3.1}$, the edge computing resource allocation can be decompotion for each EN n as follows:

$$\mathcal{P}_{3.2}: \min_{\mathbf{Y}_n} \sum_{m\in\mathcal{M}_n} \frac{g_{n,m}^{(\zeta,x)}\mu_{n,m}\gamma_{n,m}}{y_{n,m}f_n^{es}} \tag{25a}$$

$$s.t. \sum_{m\in\mathcal{M}_n} y_{n,m} \le 1, \ \forall n \in \mathcal{N} \tag{25b}$$

$$y_{n,m} \ge 0. \tag{25c}$$

The closed-form expression for the edge computing resource allocation ratio is given as follows.

$$y_{n,m} = \frac{\sqrt{g_{n,m}^{(\zeta,x)}\mu_{n,m}\gamma_{n,m}/f_n^{es}}}{\sum_{m\in\mathcal{M}_n}\sqrt{g_{n,m}^{(\zeta,x)}\mu_{n,m}\gamma_{n,m}/f_n^{es}}}. \tag{26}$$

3) Cloud Computing Resource Allocation with Fixed Task Offloading Strategy: Similar to the previous subproblem, the cloud computing resource allocation problem is decoupled into the following subproblem.

$$\mathcal{P}_{3.3}: \min_{\boldsymbol{Z}} \sum_{n\in\mathcal{N}} F_n \tag{27a}$$

$$s.t. \sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}_n} z_{n,m} \le 1, \tag{27b}$$

$$z_{n,m} \ge 0, \tag{27c}$$

where $F_n = \sum_{m\in\mathcal{M}_n} \frac{g_{n,m}^{(\theta,x)}\mu_{n,m}\gamma_{n,m}}{z_{n,m}f^{cs}}$.

To solve $\mathcal{P}_{3.3}$, the Lagrangian function of problem (27) is defined by

$$\mathcal{G}(\boldsymbol{Z},\boldsymbol{\lambda}) = \sum_{n\in\mathcal{N}} F_n + \sum_{n\in\mathcal{N}} \lambda_n\Big(\sum_{m\in\mathcal{M}_n} z_{n,m} - \frac{1}{\mathcal{N}}\Big), \tag{28}$$

where $\boldsymbol{\lambda} \in \mathcal{R}^{\mathcal{N}}$. Furthermore, for the constraint (27b) to hold, $\boldsymbol{\lambda}$ requires to reach consensus among its elements [37], i.e., the following constraint

$$\mathcal{L}\boldsymbol{\lambda} = 0, \tag{29}$$

where $\mathcal{L} \in \mathcal{R}^{\mathcal{N}\times\mathcal{N}}$ is the Laplacian matrix of a graph.

Combining (28) and (29), an new Lagrangian function is defined as follows

$$\mathcal{G}^{new}(\boldsymbol{Z},\boldsymbol{\lambda},\boldsymbol{\eta}) = \mathcal{G}(\boldsymbol{Z},\boldsymbol{\lambda}) + \boldsymbol{\eta}^\top\mathcal{L}\boldsymbol{\lambda}, \tag{30}$$

where $\boldsymbol{\eta} \in \mathcal{R}^{\mathcal{N}}$. Consequently, the optimal solution of problem (27) is equivalent to the following problem w.r.t. $\boldsymbol{Z}$[38].

$$\min_{\boldsymbol{Z}}\left\{\max_{\boldsymbol{\lambda},\boldsymbol{\eta}} \mathcal{G}^{new}(\boldsymbol{Z},\boldsymbol{\lambda},\boldsymbol{\eta})\right\}. \tag{31}$$

We solve the problem (31) using a distributed method. Each EN $n(n \in \mathcal{N})$ interacts with its neighboring nodes to acquire local information and solve its local optimization problem through a dynamic PNN. The outputs of the $\mathcal{N}$ PNNs correspond to the KKT points of the problem (27). Following the structure of PNNs in [37], the there-layer PNN for EN n is defined as follows.

$$\begin{cases} \epsilon\frac{d\boldsymbol{Z}_n}{dt} = -\boldsymbol{Z}_n + \boldsymbol{P}_\Omega(\boldsymbol{Z}_n - \nabla_{\boldsymbol{Z}_n}\mathcal{G}^{new}) \\ \epsilon\frac{d\lambda_n}{dt} = -\lambda_n + \big[\lambda_n + \mathbf{1}_{\mathcal{M}_n}^\top\boldsymbol{Z}_n - 1/\mathcal{N} \\ \qquad\qquad - \sum_{j\in\mathcal{N}_n}(\lambda_n - \lambda_j + \eta_n - \eta_j)\big]^+ \\ \epsilon\frac{d\eta_n}{dt} = \sum_{j\in\mathcal{N}_n}(\lambda_n - \lambda_j), \end{cases} \tag{32}$$

where ϵ is a time constant, $\mathbf{1}_{\mathcal{M}_n}$ is a vector of dimension $\mathcal{M}_n$ with all elements equal to 1, and

$$\nabla_{\boldsymbol{Z}_n}\mathcal{G}^{new} = \nabla F_n + \lambda_n\mathbf{1}_{\mathcal{M}_n}.$$

***Remark* 1.** *The PNNs model (32) solves problem $\mathcal{P}_{3.3}$ in a distributed manner by requiring only the change of Lagrange multiplier ($\boldsymbol{\rho}$, $\boldsymbol{\eta}$) between neighboring ENs, without sharing local date. This effectively mitigates the risk of data leakage. Moreover, the convergence of model (32) has been formally proven in [37].*

D. Task offloading scheme

This section focuses on solving problem (21) under the given fixed resource allocation strategy $\{\boldsymbol{W}^*, \boldsymbol{Y}^*, \boldsymbol{Z}^*\}$.

Let

$$\nu_n = \sum_{m\in\mathcal{M}_n}\sqrt{g_{n,m}^{(\alpha,x)}\mu_{n,m}/\Re_{n,m}^{es}}, \tag{33}$$

and

$$\omega_n = \sum_{m\in\mathcal{M}_n}\sqrt{g_{n,m}^{(\zeta,x)}\mu_{n,m}\gamma_{n,m}/f_n^{es}}. \tag{34}$$

Then, using the conditions in (16d), we derive the closed-form expressions for the task splitting ratios $\alpha_{n,m}$ and $\zeta_{n,m}$ for $n \in \mathcal{N}, m \in \mathcal{M}_n^0$ as follows.

$$\alpha_{n,m} = \frac{\sqrt{(\zeta_{n,m}+\theta_{n,m})c_{n,m}}\nu_{n,m} + \theta_{n,m}b_{n,m} + \frac{2D_n}{C}}{T_{n,m}^{loc}}, \tag{35}$$

and

$$\zeta_{n,m} = \frac{d_{n,m}(\theta_{n,m}b_{n,m} + \frac{2D_n}{C})^2}{\omega_n^2}, \tag{36}$$

where $c_{n,m} = \mu_{n,m}/\phi_{n,m}\Re_{n,m}^{es}$, $\phi_{n,m} = \phi_{n,m}^1 + \phi_{n,m}^2 P_{n,m}$, $b_{n,m} = T_{n,m}^{c,trans} + \frac{\mu_{n,m}\gamma_{n,m}}{z_{n,m}^* f^{cs}}$, $d_{n,m} = 0.5\phi_{n,m}^1 f_n^{es}/\mu_{n,m}\gamma_{n,m}$.

Using the constraints (16c), we have

$$\sqrt{(\zeta_{n,m}+\theta_{n,m})c_{n,m}}\nu_{n,m} + \theta_{n,m}b_{n,m} + \frac{2D_n}{C} + T_{n,m}^{loc}(\zeta_{n,m}+\theta_{n,m}-1) = 0. \tag{37}$$

Substituting (24) and (26) into (18), we obtain a reformulated utility function as follows.

$$\mathcal{U}_n = \nu_n^2 + \omega_n^2 + \sum_{m\in\mathcal{M}_n^0} f_{n,m}^{(\alpha,\theta)} + \sum_{m\in\mathcal{M}_n^1} f_{n,m}^{(x)} + \sum_{m\in\mathcal{M}_n} \frac{g_{n,m}^{(\theta,x)}\mu_{n,m}\gamma_{n,m}}{z_{n,m}^* f^{cs}}. \tag{38}$$

It follows from (35) and (36) that the expression of ν_n and ω_n in (33) and (34) can be rewritten as follows.

$$\begin{aligned}\nu_n - &\sum_{m\in\mathcal{M}_n^0}\phi_{n,m}\sqrt{(\zeta_{n,m}+\theta_{n,m})c_{n,m}}\\ &-\sum_{m\in\mathcal{M}_n^1}\phi_{n,m}x_{n,m}(3-2x_{n,m})\sqrt{c_{n,m}}=0,\end{aligned}\tag{39}$$

and

$$\begin{aligned}2\omega_n^2 - &\sum_{m\in\mathcal{M}_n^0}\left(\theta_{n,m}b_{n,m}+\frac{2D_n}{C}\right)\phi_{n,m}^1\\ &-2\omega_n\sum_{m\in\mathcal{M}_n^1}4x_{n,m}(1-x_{n,m})\frac{\phi_{n,m}^1}{\sqrt{2d_{n,m}}}=0.\end{aligned}\tag{40}$$

Consequently, we reformulate problem (21) as follows.

$$\mathcal{P}_5: \quad \min_{\boldsymbol{\theta}_n,\nu_n,\omega_n,\boldsymbol{X}_n} \mathcal{U}_n \tag{41a}$$

$$s.t. \quad (16b), (21c), (37), (39), (40) \tag{41b}$$

To solve subproblem $\mathcal{P}_5$, we employ the penalty function method to effectively handle the constraints. Let $\boldsymbol{u}_n = [\boldsymbol{\theta}_n^\top; \boldsymbol{X}_n^\top; \nu_n; \omega_n]$ be the decision variable vector, and $g_{n,m}$ denote the equality constraints in the original problem. Then, problem $\mathcal{P}_5$ can be reformulated into an unconstrained optimization problem as follows.

$$\mathcal{L}_n^\beta = \mathcal{U}_n + \frac{\beta}{2}\sum_{m\in\mathcal{L}_n^{sum}} g_{n,m}^2, \tag{42}$$

where $\mathcal{L}_n^{sum} = \mathcal{M}_n + 2$, and β is a positive constant.

Problem $\mathcal{P}_5$ is non-convex optimization problem, making it challenging to find the optimal solution. In addition, solving problem $\mathcal{P}_5$ is equivalent to solving its unconstrained reformulation in (42). Here, we employ the framework of CNO to solve (42) efficiently. $\mathcal{J}(j = 1, 2, \cdots, J)$ groups of recurrent neural networks (RNNs) are constructed, and their dynamic equations [1] are defined as follows.

$$\epsilon\frac{d\boldsymbol{u}_n^{(j)}}{dt} = -\nabla\mathcal{U}_n(\boldsymbol{u}_n^{(j)}) - \beta\nabla\boldsymbol{g}_n(\boldsymbol{u}_n^{(j)})\circ\boldsymbol{g}_n(\boldsymbol{u}_n^{(j)}), \tag{43}$$

where $\boldsymbol{g}_n = [g_{n,1}, g_{n,2}, \cdots, g_{n,m}], m \in \mathcal{L}_n^{sum}$.

We apply the PSO rule to reset all the initial values of (43), which is given by

$$\begin{aligned}\boldsymbol{v}^{(i+1,j)} =& \mathcal{W}v^{(i,j)} + c_1r_1(\boldsymbol{u}_p^{(i,j)} - \boldsymbol{u}^{(i,j)})\\ &+ c_2r_2(\boldsymbol{u}_g^{(i)} - \boldsymbol{u}^{(i,j)}),\\ \boldsymbol{u}_0^{(i+1,j)} =& \Im(\boldsymbol{u}^{(i,j)} + \boldsymbol{v}^{(i+1,j)}),\end{aligned}\tag{44}$$

for each RNN j, where $c1$ and $c2$ are two positive acceleration constants, $r1, r2 \in [0, 1]$ are two random numbers, and $\mathcal{W} \in [0, 1]$ is the inertia weight. $\boldsymbol{u}^{(j)} \in \mathcal{R}^{\sum_{n\in\mathcal{N}}\mathcal{L}_n^{sum}}$ and $\boldsymbol{v}^{(j)} \in \mathcal{R}^{\sum_{n\in\mathcal{N}}\mathcal{L}_n^{sum}}$ are the position and the velocity of the jth RNN, respectively. During the ith iteration, the jth RNN identifies its best-known position as $\boldsymbol{u}_p^{(i,j)}$, while the collective best position of the entire RNN group is denoted as $\boldsymbol{u}_g^{(i)}$. In addition, let $\overline{\boldsymbol{u}}^{(i,j)}$ and $\boldsymbol{u}_0^{(i,j)}$ be the equilibrium point and the initial state of RNN j at the ith iteration, the output vector of RNN j is defined by

$$\boldsymbol{u}^{(i,j)} := \Im(\overline{\boldsymbol{u}}^{(i,j)}).$$

On the other hand, in order to enhance the exploration capability of the algorithm and prevent premature convergence to local optima, we introduce the levy mutation operator as follows

$$\boldsymbol{u}^{(i,j)} \leftarrow \boldsymbol{u}^{(i,j)} + \boldsymbol{\chi}_\ell, \tag{45}$$

where $\boldsymbol{\chi}_\ell$ is a random vector generated by the levy distribution [40]. The mutation operation (45) is activated when the diversity indicator $N(i)$ is falls below a preset threshold τ_0. The diversity indicator, which quantifies the variance among candidate solutions, is defined as.

$$N(i) = \frac{1}{\mathcal{J}}\sum_{j\in\mathcal{J}}\|\boldsymbol{u}_p^{(i,j)} - \boldsymbol{u}_g^{(i,j)}\|. \tag{46}$$

E. Joint Optimization of Task Offloading and Resource Allocation

CNO-TORA is proposed to jointly optimize mixed task offloading strategies and resource allocation by an iterative approach, as outlined in Algorithm 1. Steps 2-26 are the outer loop, and steps 3-11 are the inner loop. In each inner loop, multiple RNNs are employed to perform parallel local searches for optimal task offloading strategies. Given these strategies, the PNN model (32) is then used to update the cloud resource allocation scheme. Steps 8-10 identify the best individual solution obtained in the current iteration. Subsequently, steps 12–14 update the global best solution based on the selected best candidate solutions. In steps 15-25, the initial values of RNNs (43) are redefined by using the PSO rule in (44). This iterative process continues until convergence is achieved. The schematic diagram of the CNO-TORA is presented in Fig. 3.

Remark 2. *It should be noted that the PNN model in (32) and the RNN model in (43) are essentially ordinary differential equations (ODEs). As such, they can be efficiently solved using standard numerical methods, such as the Euler method, the fourth-order Runge–Kutta method, or MATLAB's built-in solvers including ode15s, ode45, and ode23. The equilibrium solutions of these dynamical systems correspond to the optimal solutions of the original optimization problems.*

F. Scalability Analysis

Here, we analyze the scalability of our distributed resource allocation and task offloading framework from the perspective of the dimensionality of optimization variables. In a centralized approach [30], solving (27) and (41) requires a centralized node to handle the optimization problem with a variable dimension of $2(\mathcal{M}+\mathcal{N})+1$, where $\mathcal{M}$ and $\mathcal{N}$ represent the total number of MDs and ENs, respectively. In contrast, our distributed scheme allows each EN to independently optimize for its locally connected MDs, significantly reducing the dimensionality of the optimization problem. In

[1]The equilibrium point output of the RNN dynamic equation (43) corresponds to a local optimization of problem $\mathcal{P}_5$. For a detailed theoretical proof, please refer to [39].

Algorithm 1: CNO-TORA

Input: All computation task $q_{n,m}$, utility weights $\ell_{n,m}$, latency and energy trade-off weights $\delta^1_{n,m}$ and $\delta^2_{n,m}$, $n \in \mathcal{N}, m \in \mathbb{M}_n$, network configuration parameters, algorithm parameters, Laplacian matrix $\mathcal{L}$

Output: The system solution $\boldsymbol{Z}^*$,$\boldsymbol{u}^*$ and the corresponding utility U

Initialization: Set initial state $\boldsymbol{u}_0^{(1,j)}$, velocity $\mathbf{v}_0^{(1,j)}$, $\boldsymbol{u}_p^{(0,j)} \leftarrow \boldsymbol{u}_0^{(1,j)}$,$\boldsymbol{Z}_p^{(0,j)} \leftarrow \boldsymbol{Z}_0^{(1,j)}$, $\boldsymbol{u}_g^{(0)}$, $\boldsymbol{Z}_g^{(0)}$, $\mathcal{W}$, c_1, c_2, $k \leftarrow 0$, $i \leftarrow 1$, τ_0, δ, $\mathcal{L}_{max}$ and k_{max},

while $i \leq \mathcal{L}_{max}$ && $k \leq k_{max}$ **do**
  for $j = 1 : \mathcal{J}$ **do**
    for $n \in \mathcal{N}$ **do**
      Calculate the output $\boldsymbol{u}^{(i,j)} = \Im(\overline{\boldsymbol{u}}^{(i,j)})$ of each RNN using Eq. (43) with initial values $\boldsymbol{u}_0^{(i,j)}$;
    end
    Compute cloud computing resource allocation using Eq. (32);
    if $U(\boldsymbol{Z}^{(i)}, \boldsymbol{u}^{(i,j)}) \geq U(\boldsymbol{Z}_p^{(i-1)}, \boldsymbol{u}_p^{(i-1,j)})$ **then**
      $\boldsymbol{u}_p^{(i,j)} \leftarrow \boldsymbol{u}^{(i,j)}$;$\boldsymbol{Z}_p^{(i)} \leftarrow \boldsymbol{Z}^{(i)}$;
    end
  end
  if $\max_{j\in\mathcal{J}} U(\boldsymbol{Z}_p^{(i)}, \boldsymbol{u}_p^{(i,j)}) \geq U(\boldsymbol{Z}_g^{(i-1)}, \boldsymbol{u}_g^{(i-1,j)})$ **then**
    $\boldsymbol{Z}_g^{(i)} \leftarrow \arg\max_{j\in\mathcal{J}} U(\boldsymbol{Z}_p^{(i)}, \boldsymbol{u}_p^{(i,j)})$;
    $\boldsymbol{u}_g^{(i)} \leftarrow \arg\max_{j\in\mathcal{J}} U(\boldsymbol{Z}_p^{(i)}, \boldsymbol{u}_p^{(i,j)})$;
  end
  Compute the diversity metric $N(i)$ by (46).
  if $N(i) \leq \tau_0$ **then**
    $\boldsymbol{u}^{(i,j)} \leftarrow \boldsymbol{u}^{(i,j)} + \boldsymbol{\chi}_\ell$;
  end
  Update the initial states $\boldsymbol{u}_0^{(i+1,j)}$ based on Eq. (44)
  if $|\boldsymbol{u}_g^{(i)} - \boldsymbol{u}_g^{(i-1)}| \leq \delta$ **then**
    $k \leftarrow k + 1$;
  else
    $k \leftarrow 0$;
  end
  $i \leftarrow i + 1$
end
Return $\boldsymbol{Z}^* = \boldsymbol{Z}_g^{(i)}$, $\boldsymbol{u}^* = \boldsymbol{u}_g^{(i)}$ and $U(\boldsymbol{Z}^*, \boldsymbol{u}^*)$

this case, the variable dimension at EN n is $2(\mathcal{M}_n + \mathcal{N}) + 2$, where $\mathcal{M}_n$ denotes the number of MDs associated with EN n. Thus, the proposed distributed algorithm is highly scalable and efficient for the optimization problems when $2(\mathcal{M}_n + \mathcal{N}) + 2 \ll 2(\mathcal{M} + \mathcal{N}) + 1$ i.e., the system has a large number of MDs but also a sufficient number of ENs to distribute the computational load efficiently. This ensures that as the network size increases, each EN only handles a fraction of the overall problem, significantly reducing the complexity compared to a centralized optimization scheme.

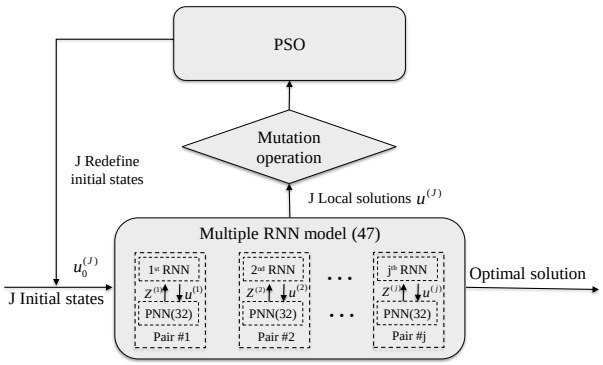


Fig. 3. A schematic diagram of the CNO-TORA algorithm.

G. Convergence Analysis

Now, we analyze the convergence of the Algorithm 1. The original optimization problem in Algorithm 1 is given by

$$\max_{(\boldsymbol{Z},\boldsymbol{u})\in\Omega} U(\boldsymbol{Z}, \boldsymbol{u}), \tag{47}$$

where $\Omega = \Omega_z \times \Omega_u$, Ω_z a convex set and Ω_u a nonconvex set. For every fixed $\boldsymbol{u} \in \Omega_u$, the subproblem

$$\max_{\boldsymbol{Z}\in\Omega_z} U(\boldsymbol{Z}, \boldsymbol{u}) \tag{48}$$

is convex and hence can be solved to global optimality. Let

$$\boldsymbol{Z}^*(\boldsymbol{u}) = \operatorname*{argmax}_{\boldsymbol{Z}\in\Omega_z} \mathcal{U}(\boldsymbol{Z}, \boldsymbol{u})$$

denote the unique (or globally optimal) solution of (48). Then, the original problem (47) is equivalent to the reduced problem

$$\max_{\boldsymbol{u}\in\Omega_u} \widetilde{U}(\boldsymbol{u}), \tag{49}$$

where $\widetilde{U}(\boldsymbol{u}) = U\big(\boldsymbol{Z}^*(\boldsymbol{u}), \boldsymbol{u}\big)$. Consequently, we obtain the following objective function sequence.

$$\widetilde{U}\left(\boldsymbol{u}_g^{(i+1)}\right) = \begin{cases} \widetilde{U}\left(\boldsymbol{u}_g^{(i)}\right), & \text{if } \widetilde{U}\left(\boldsymbol{u}_p^{(i+1)}\right) \leq \widetilde{U}\left(\boldsymbol{u}_g^{(i)}\right), \\ \widetilde{U}\left(\boldsymbol{u}_p^{(i+1)}\right), & \text{otherwise.} \end{cases}$$

It is evident that the sequence $\{\widetilde{U}(\boldsymbol{u}_g^{(i)})\}_{i=1}^{+\infty}$ is monotonically non-decreasing. In addition, from (17), we obtain

$$\widetilde{U}\left(\boldsymbol{u}_g^{(i)}\right) \leq \sum_{n\in\mathcal{N}} \sum_{m\in\mathcal{M}_n} \left(\phi^1_{n,m} T^{loc}_{n,m} + \phi^2_{n,m} E^{loc}_{n,m}\right) < +\infty.$$

By the Bolzano–Weierstrass theorem [41], the objective function sequence $\{\widetilde{U}(\boldsymbol{u}_g^{(i)})\}_{i=1}^{+\infty}$ is convergent.

Let $\Theta_{i,j}$ be the supporting set of the initial states of RNNs (43) at the ith-iteration. Since the initial states of RNNs (43) are mapped in the the feasible set Ω_u by the second update rules in (44), we obtain $\Omega_u \subseteq \Theta := \bigcup_{i=1}^{K_s} \bigcup_{j=1}^{\mathcal{J}} \Theta_{i,j}$ $(1 \leq K_s)$. Consequently, the Lebesgue measure satisfies $\mathcal{L}(\Omega_u) = \mathcal{L}(\Theta) > 0$. By [42, Th. 1], it follows that

$$\lim_{i\to\infty} P\left(\boldsymbol{u}_g^{(i)} \in \Omega_u^g\right) = 1,$$

where Ω_u^g is the set of the global optimal solutions. This proves that the CNO-TORA globally converges to a global optimal solution of problem (41) with probability one.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2025.3599885

TABLE III
PARAMETER SETTING

Network Parameter	**Setting**
Input data-size, $\mu_{n,m}$	$[0.5, 1]$ Mbits
Workload of one-bit date, $\gamma_{n,m}$	1000 CPU cycle/bits
Maximum transmission power, $P_{n,m}$	30 dBm
Noise power, σ_n	-174 dBm/Hz
Path loss between MDs and ENs,	$41 + 28log10(d)$ dB
Bandwidth, B_n	$[5, 30]$MHz
Backhaul link capacity, $\Re_n^{cs}$	20 Mbps
Device computation capacity, $f_{n,m}^{loc}$	$[0.5, 3]$ GHz
Edge computation capacity, f_n^{es}	$[20, 80]$ GHz
Cloud computation capacity, f^{cs}	$[50, 300]$ GHz
Algorithm Parameter	**Setting**
Preset threshold, τ_0	0.2
Error tolerance, δ	10^{-6}
Time constant, ϵ	10^{-6}
Termination criteria, $\mathcal{L}_{max}$, k_{max}	$\mathcal{L}_{max} = 20$, $k_{max} = 5$

TABLE IV
EXPERIMENTAL SETTING

		$\mathcal{M}$	$f_{n,m}^{loc}$	$\mathcal{N}$	B_n	f_n^{es}	f^{cs}
Set ♯1	♯1.1	40	1	4	10	20	100
	♯1.2	[20,400]	1	10	10	20	100
Set ♯2	♯2.1	$[20, 60]$	1	4	10	20	100
	♯2.2	40	$[0.5, 2.5]$	4	10	20	100
	♯2.3	40	1	[2,6]	10	20	100
	♯2.4	40	1	4	[5,25]	20	100
	♯2.5	40	1	4	10	$[10, 50]$	100
	♯2.6	40	1	4	10	20	[50,250]

H. Complexity Analysis

In this section, we analyze the computational complexity of the proposed CNO-TORA. In Algorithm 1, the complexity is influenced by the number of RNNs $\mathcal{J}$, the number of iterations $\mathcal{L}_{max}$, the number of ENs $\mathcal{N}$, and the number of MDs $\mathcal{M}$. Consequently, the total complexity of lines 3-6 is given by $\mathcal{O}(\mathcal{J}\mathcal{L}_{max}(\mathcal{M} + 2\mathcal{N}))$. Additionally, solving the distributed PNN model (32) for cloud resource allocation incurs a complexity of $\mathcal{O}(\mathcal{L}_{max}\widehat{\mathcal{L}}_c(\mathcal{M}+2\mathcal{N}))$, where $\widehat{\mathcal{L}}_c$ is the number of iterations required for convergence. By combining both algorithms, the overall computational complexity of the Algorithm 1 is given by $\mathcal{O}(\mathcal{L}_{max}(\widehat{\mathcal{L}}_c + \mathcal{J})(\mathcal{M} + 2\mathcal{N}))$.

VI. EVALUATION

In this section, we evaluate the performance of the proposed algorithm through simulation experiments. All simulations were performed on a personal laptop equipped with an *AMD Ryzen 7 5800H*. The algorithm was implemented in *MATLAB R2022b*, and *ode15s* solver and *Parallel Computing Toolbox* are used.

Throughout our simulation experiments, the communication topology between ENs is an undirected connected ring graph. Within the coverage area, several MDs are randomly distributed and served by the corresponding EN over the wireless channel. In the network environment, we assume that 80% of users perform full offloading, while 20% perform partial offloading. The data size of the computation task follows a uniform distribution, i.e., $\mu_{n,m} \in [0.1, 0.5]$ [Mbits] and the required CPU cycles per bit is set by $\gamma_{n,m} = 1000$ [CPU cycle/bits]. The other settings of the simulation parameters are given in Table III.

Baselines. The baselines used in this paper are listed as follows.

- *Random offloading*: Computation tasks are randomly split and offloaded to local devices, ENs, and CS for processing. Then, the proposed iterative approach and resource allocation scheme are applied to solve the formulated problem.
- *Greedy offloading*: Indivisible tasks are offloaded to local, ES, or CS for processing based on the user's maximum utility, while divisible tasks follow a random task-splitting ratio. The proposed iterative approach and resource allocation scheme are then applied to solve the formulated problem.
- *Adaptive PSO (APSO)* [43]: A modified adaptive PSO algorithm with nonlinear inertia weight to optimize edge-cloud collaborative offloading strategy.
- *Hybrid Genetic-Simulated Annealing-PSO (GSP)* [32]: A meta-heuristic algorithm combining genetic algorithm (GA), simulated annealing (SA), and PSO for task offloading and resource allocation optimization.
- *Centralized LEOS-Assisted Offloading (CLO)* [30] A centralized alternating optimization strategy that optimizes task offloading and resource allocation in a LEOS edge-assisted MEC system.

Dataset. A real dataset [44] obtained from the Central Business District (CBD) of Melbourne, Australia, is used in all our experiments. It covers a total area of $6.2km^2$ and includes latitude and longitude information for 816 MDs and 125 BSs.

Experiment Settings. In each experiment, we randomly extracted some ESs and MDs from the dataset to simulate an MEC system. For simplicity, we set the ESs to have the same computing and bandwidth resources. To comprehensively evaluate the effectiveness of our proposed approach, we simulate various MEC systems by varying the following parameters in the experiments: 1) the number of MDs ($\mathcal{M}$); 2) the number of ESs ($\mathcal{N}$); 3) the computing resources on ESs (f_n^{es}); 4) the bandwidth resources (B_n); 5) the computing resources on CSs (f^{cs}); and 6) the computing capacity of MDs ($f_{n,m}^{loc}$). The main parameter settings are summarized in Table IV. Unless otherwise stated, each experiment is repeated 20 times. The number of RNNs is set to $\mathcal{J} = 4$; the utility parameters are set to $\ell_{n,m} = 1$; and the latency and energy trade-off parameters are set to $\delta_{n,m}^1 = \delta_{n,m}^2 = 0.5$.

A. Algorithm Convergence and Effectiveness Analysis

Convergence and constraint verification. Here, we first verify the convergence of the proposed algorithm including PNN model (32), RNN model (43) and Algorithm 1, the results are shown in Figs 4-6. These results validate the convergence and effectiveness of the proposed algorithm.

In Set ♯1.1, Fig. 4 shows the convergence behavior of Algorithm 1 with respect to iterations. As observed from the figure, Algorithm 1 converge as the number of iterations increases, thus validating the theoretical analysis presented in this paper.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2025.3599885

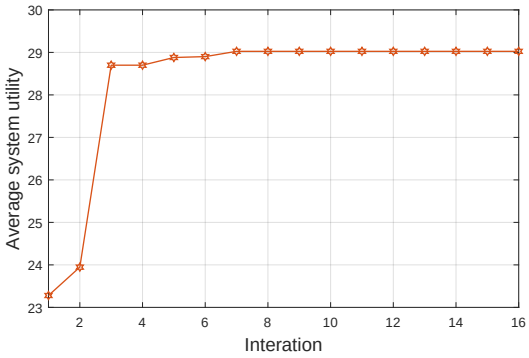


Fig. 4. Convergent behavior of the objective function values obtained by using Algorithm 1. (Set #1.1)

Fig. 5 and Fig. 6 illustrate the time behavior of the decision variable and the Lagrangian multipliers. From the figures, all states of models (32) and (43) converge to their equilibrium states, which indicates that the model (32) and model (43) are convergent. Furthermore, Figs. 5(a), 5(b), 5(d), and 5(e) depict the cloud computing resource allocation ratios to different ENs users. The total sum of these ratios is 1.001, which is slightly higher than 1 due to the numerical precision limits. However, this minor deviation remains within an acceptable tolerance and does not significantly affect the validity of the inequality constraint (16h). Fig. 5(c) verifies that as the number of iterations increases, the Lagrangian multipliers $\lambda_n(n \in \mathcal{N})$ all converge to the same value, thereby satisfying the constraint (29). Since $\eta_n(n \in \mathcal{N})$ are unconstrained Lagrangian multipliers, they can take both positive and negative values, which is verified in Fig. 5(f). The constraints in problem $\mathcal{P}_{3.3}$ are fulfilled. In addition, the decision variables $x_{n,m}(n \in \mathcal{N}, m \in \mathcal{M}_n)$ in Fig. 6 converge to 0, 0.5, or 1 over time, satisfying the constraint (19b). Additionally, the cloud task splitting ratios $\theta_{n,m}(n \in \mathcal{N}, m \in \mathcal{M}n)$ remain within the range $0 < \theta_{n,m} < 1$, and both ν_n and $\omega_n(n \in \mathcal{N})$ are positive. These observations confirm that all constraints in $\mathcal{P}_5$ are satisfied.

Time complexity analysis. Fig. 7 evaluates how the computation time of the proposed CNO-TORA algorithm varies with system parameters, namely the number of RNNs $\mathcal{J}$, MDs $\mathcal{M}$, and ENs $\mathcal{N}$. Overall, the observed trends show that the computation time increases approximately linearly with respect to each of these parameters. This behavior is consistent with the theoretical complexity analysis presented in Subsection V-H.

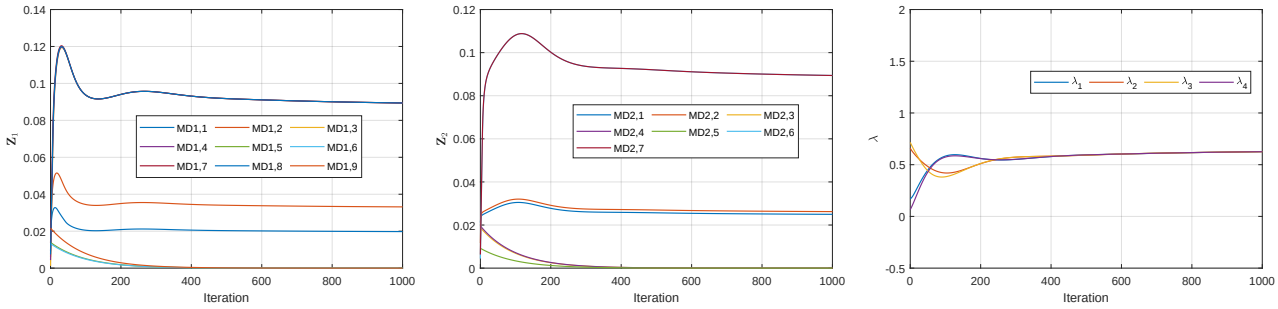
i) Impact of RNN number $\mathcal{J}$. In Fig. 7(a), as the number of RNNs increases from 2 to 12, computation time increases nearly linearly with the number of RNNs, which confirms the $\mathcal{O}(\mathcal{J})$ linear dependency in the theoretical complexity. This trend is expected, as a larger number of RNNs enables more parallel local searches per iteration, thereby increasing the overall computational load. Meanwhile, the average system utility also improves with increasing $\mathcal{J}$, as more RNNs enhance the diversity of solution exploration and help identify higher-quality solutions, which can be confirmed in Fig. 7(a). Therefore, this trade-off offers practical flexibility: fewer RNNs yield faster decisions, while more RNNs lead to better performance. Fig.7(b) shows that the resource utilization ratio slightly decreases with increasing $\mathcal{J}$. This is due to the fact that in small-scale user scenarios, fully offloading tasks to the edge or cloud is often more beneficial. With more RNNs, the offloading strategies become more accurate, and some tasks that were previously executed locally are now offloaded to the edge or cloud for better performance. This shift reduces the usage of local device resources, resulting in a slight drop in overall resource utilization.

ii) Impact of MDs Number $\mathcal{M}$. As shown in Fig.7(c), computation time increases steadily as the number of MDs $\mathcal{M}$ changes from 40 to 120. This trend arises because each additional MD introduces a fixed number of decision variables and constraints related to task offloading and resource allocation. Consequently, the computational load per iteration increases proportionally, and more iterations are required for convergence. Fig. 7(d) shows a non-monotonic trend in resource utilization as $\mathcal{M}$ increases; it first decreases, then increases, and finally decreases again. This behavior can be interpreted as follows. When the number of MDs is small, most tasks benefit from being offloaded to the edge or cloud, which reduces local resource usage and leads to lower utilization ratios. As $\mathcal{M}$ increases, resource contention makes offloading less beneficial, especially when $\mathcal{M} > 100$, leading more users to prefer local execution. This increases the usage of local resources and improves overall utilization. However, as the number of MDs continues to grow, the increase in locally executed tasks cannot keep pace with the growing total number of MDs. Consequently, the average resource utilization ratio begins to decline again.

iii) Impact of ENs Number $\mathcal{N}$. From Fig. 7(e), we observe that the computation time increases approximately linearly with the number of ENs, which is consistent with the complexity analysis of the proposed algorithm. Although increasing the number of ENs can reduce the number of optimization variables, every EN still needs to execute the optimization algorithm. The time saved per EN is insufficient to offset the additional computation introduced by the ENs. As a result, the overall computation time grows. Fig. 7(f) shows that the added ENs are effectively utilized, as seen in the linear growth in resource utilization, indicating that computational resources are not idle despite higher complexity.

Scalability verification. To evaluate the scalability of Algorithm 1, we analyze its performance under large-scale networks scenario, where the number of ENs is set to $\mathcal{N} = 10$, and the number of MDs, $\mathcal{M}$, varies from 100 to 400. The remaining network parameters follow the settings in Set #1.2. Specifically, we measure two critical metrics: (i) the average system utility achieved and (ii) the computation time required for the optimization process. The results are illustrated in Figure 8. From the figure, we observe that the average system utility (blue solid line) increases as the number of MDs grows from 100 to 400, reaching a peak value. Beyond this point, the utility starts to decrease, suggesting that resource contention leads to system utility degradation. Meanwhile, the computation time (orange dashed line) steadily increases as the number of MDs grows. This trend is attributed to the

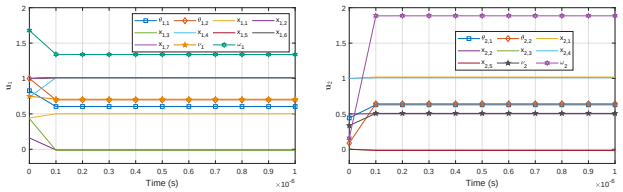
This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2025.3599885



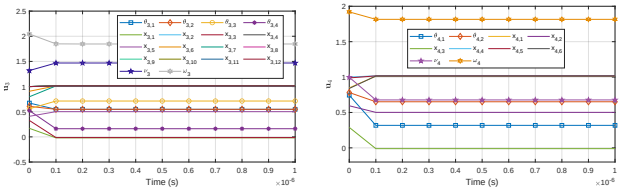
(a) Cloud resource allocation ratio at EN 1. (b) Cloud resource allocation ratio at EN 2. (c) Lagrangian multipliers $\boldsymbol{\lambda}$.

(d) Cloud resource allocation ratio at EN 3. (e) Cloud resource allocation ratio at EN 4. (f) Lagrangian multipliers $\boldsymbol{\eta}$.

Fig. 5. The state convergence behavior of the PNN model (32) (Set ♯1.1).



(a) The decision variable of EN 1. (b) The decision variable of EN 2.



(c) The decision variable of EN 3. (d) The decision variable of EN 4.

Fig. 6. The state convergence behavior of the RNN model (43) (Set ♯1.1).

expansion of the problem size. However, the growth remains moderate, indicating that the algorithm is computationally feasible for large-scale networks.

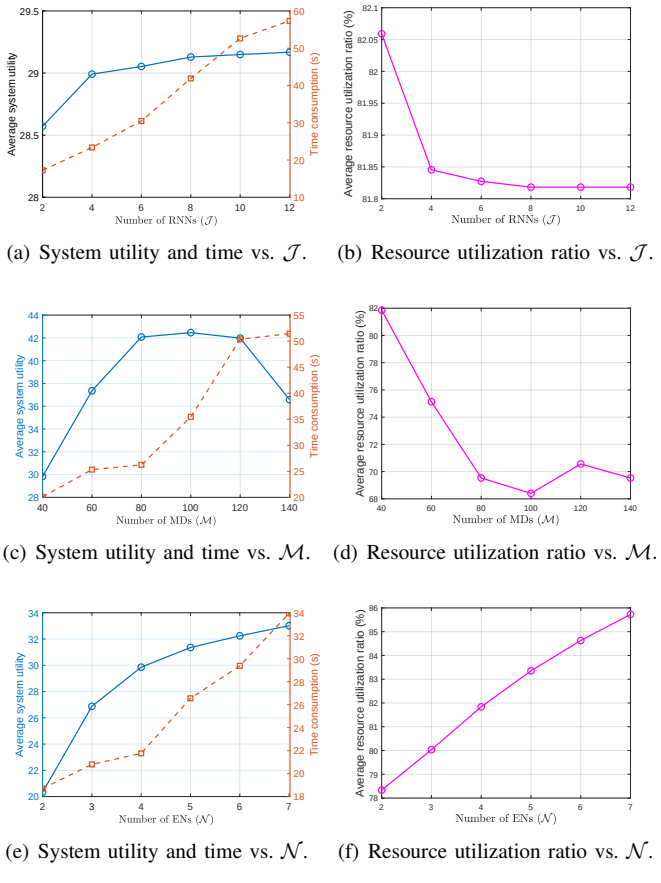
Performance Comparison. Table V presents the statistical results of the average system utility obtained by Algorithm 1 and five baseline methods over 100 independent runs under different numbers of MDs. As shown in the table, our proposed distributed algorithm outperforms all baselines in terms of best-case, worst-case, average utility, and standard deviation (SD), demonstrating both superior performance and robustness. Regarding computation time, our method is more efficient than CLO, which indicates that our distributed algorithm can achieve high solution quality with lower computational overhead. Overall, our algorithm exhibits robust and stable performance, achieving improvements in average system utility of 15.71%-16.59%, 14.88%-16.18%, 13.28%-17.45%, 19.6%-28.15%, and 4.01%-10.71% compared to the Random, Greedy, APSO, GSP, and CLO, respectively.

These superior results can be explained by the differences in decision-making strategies. Random and Greedy algorithms make decisions over the local feasible domain without in-depth exploration of the global solution structure. Although fast, they are prone to local fluctuations, resulting in lower utility. In contrast, the heuristic algorithms APSO and GSP search for the global optimum across the entire feasible domain, but their relatively crude search strategies often lead them to become trapped in local optima, causing unsatisfactory worst-case utility. The CLO method employs centralized optimization, which ensures relatively stable results (i.e., low SD), but its centralized nature leads to longer computation times. In contrast, our proposed method outperforms all baselines thanks to two key design points. First, the CNO-TORA algorithm adopts a strategy of searching for the global optimum from the local optima, enabling it to lock onto high-utility solutions more quickly. Second, we designed a distributed convex optimization algorithm based on the PNN model, which efficiently allocates cloud computing resources. Thus, our approach not only fully exploits local advantages but also guarantees effective global resource coordination, thereby achieving high system utility with stability.

B. Parameter Analysis

Fig. 9 illustrate the impact of the six parameters set in Table IV including $\mathcal{M}$, $\mathcal{N}$, B_n, f_n^{es}, f^{cs} and $f_{n,m}^{local}$ on the average system utility compared to five baseline schemes. Overall, it

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2025.3599885



(a) System utility and time vs. $\mathcal{J}$. (b) Resource utilization ratio vs. $\mathcal{J}$.

(c) System utility and time vs. $\mathcal{M}$. (d) Resource utilization ratio vs. $\mathcal{M}$.

(e) System utility and time vs. $\mathcal{N}$. (f) Resource utilization ratio vs. $\mathcal{N}$.

Fig. 7. The resource utilization ratio, average system utility, and computation time achieved by CNO-TORA under different numbers of RNNs, MDs, and ENs in Sets #1.1, #1.2 and #2.3, respectively (averaged over 100 runs).

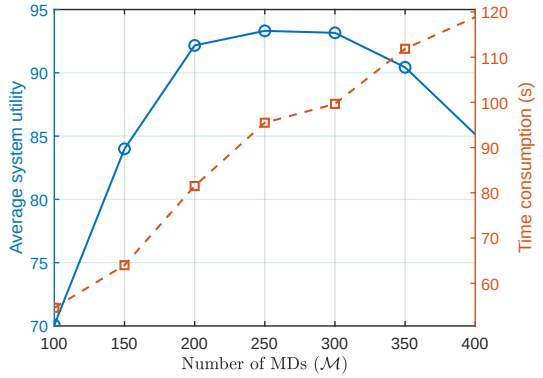


Fig. 8. The average system utility obtained by Algorithm 1 and the computation time for different numbers of MDs. (Set #1.2)

is evident that our scheme outperforms all other approaches, improving the system utility by an average of 10.39% to 23.85% over the baselines in Set ♯ 2.

In Set ♯2.1, Fig. 9(a) shows the average system utility versus different MDs ($\mathcal{M}$). When the number of MDs is relatively small, offloading tasks to ENs and CS provides considerable benefits. However, as the number of MDs increases, the system utility gradually diminishes and eventually reaches a saturation point, beyond which local processing at MDs becomes more beneficial. This phenomenon mainly

TABLE V
STATISTICS OF AVERAGE SYSTEM UTILITY OBTAINED BY ALGORITHM 1 AND FIVE BASELINES UNDER DIFFERENT NUMBERS OF MDS (SET ♯1.2).

Test Item	Method	Type	Best	Worst	Average	SD	Time (s)
$\mathcal{M}=20$	Random	Dis.[2]	16.9301	12.1958	14.6767	0.6830	8.3786
	Greedy	Dis.	16.7898	12.5284	14.7161	0.6263	17.2986
	APSO	Cen.[1]	16.7434	13.4959	15.0928	0.4730	**0.0273**
	GSP	Cen.	15.9643	12.7793	14.3606	0.5078	0.5247
	CLO	Cen.	16.8277	15.7659	16.4388	0.0557	19.7801
	Proposed	Dis.	**17.2270**	**16.3595**	**17.0977**	**0.0142**	14.7041
$\mathcal{M}=30$	Random	Dis.	21.5604	17.3754	19.4940	0.7472	9.3975
	Greedy	Dis.	21.6142	17.4668	19.6112	0.6765	18.6369
	APSO	Cen.	21.3247	17.8395	19.4805	0.4232	**0.0274**
	GSP	Cen.	19.9130	16.5774	18.1963	0.4387	0.6888
	CLO	Cen.	22.1315	18.6949	20.8976	0.4463	27.4681
	Proposed	Dis.	**23.0055**	**22.0053**	**22.7277**	**0.0386**	19.0901
$\mathcal{M}=40$	Random	Dis.	27.6240	22.2394	25.0700	0.8543	10.0315
	Greedy	Dis.	27.8968	23.2725	25.2501	0.7998	20.9191
	APSO	Cen.	28.2969	22.8140	24.6928	0.8200	**0.0284**
	GSP	Cen.	24.6616	20.4033	22.6353	0.8340	0.9073
	CLO	Cen.	27.5811	24.4013	26.2022	0.5414	29.2014
	Proposed	Dis.	**29.3038**	**28.2291**	**29.0082**	**0.0294**	21.6583

[1] Indicates centralized algorithm.
[2] Indicates distributed algorithm.

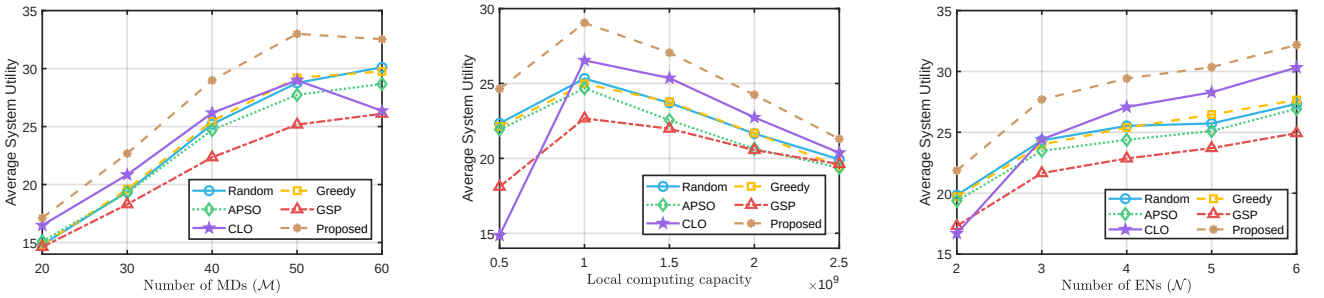
arises because an increased number of MDs intensifies the competition for computing and wireless resources at ENs and CS, resulting in higher overhead for task transmission and processing. This, in turn, reduces the offloading utility. Fig. 9(b) shows a similar trend with the increase in local capacity. The primary reason for this behavior is that as local computational resources increase, more tasks can be processed locally without the need for offloading to ENs or CS. Compared to the baseline methods, our proposed approach achieves the best performance, providing improvements ranging from 12.83% to 25.36% in Set ♯2.1, and 10.39% to 28.44% in Set ♯2.2.

In Set ♯2.3, Fig. 9(c) shows the average system utility versus different ENs ($\mathcal{N}$). As the number of ENs increases, the total computational capacity and bandwidth resources of the system also expand, which enables more resources to be allocated to MDs for their computation task offloading. This enables MDs to offload their tasks more effectively, benefiting from reduced processing delays and lower energy consumption. Consequently, the system utility improves for all approaches. Compared to baseline methods, our proposed scheme achieves the highest utility, with an improvement in average system utility ranging from 15.03% to 27.89%.

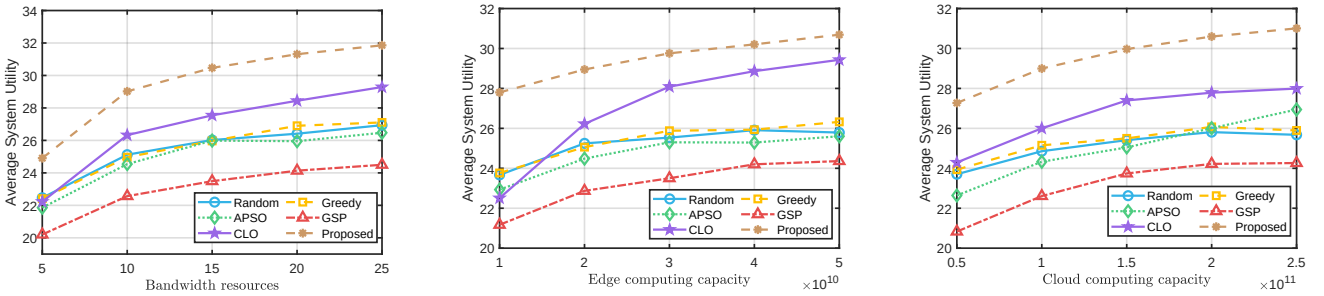
In Set ♯2.4, Fig. 9(d) illustrates the average system utility versus bandwidth resources (B_n). Initially, increasing bandwidth leads to a significant improvement in system utility. This is because greater bandwidth allows more task offloading, reducing transmission delays and enabling more MDs to utilize ENs and CS resources. However, as bandwidth continues to increase, the rate of utility improvement gradually slows. The primary reason for this trend is that as more tasks are offloaded, computing resources gradually become the new bottleneck, which limits system performance improvement. In addition, compared to baselines, our scheme achieves at least 16.27% to 28.44% average system utility improvement.

In Set ♯2.5 and ♯2.6, Fig. 9(e) and Fig. 9(f) show the average system utility versus the computing capacities of ENs (f_n^{es}) and CS (f^{cs}), respectively. Both figures follow a similar trend to Fig. 9(d), where increasing computing capacity initially leads to a significant improvement in system utility. However,

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2025.3599885



(a) System utility vs. Number of MDs (b) System utility vs. Local computing capacity (c) System utility vs. Number of ENs



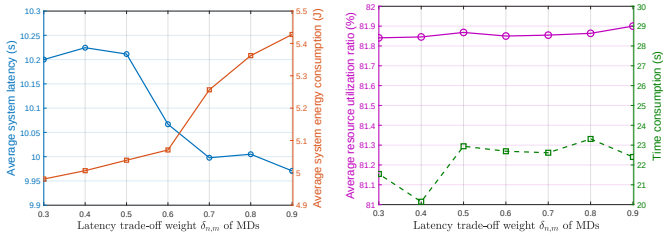
(d) System utility vs. Bandwidth resources (e) System utility vs. Edge computing capacity (f) System utility vs. Cloud computing capacity

Fig. 9. Average system utility versus the impact of different parameters (Set ♯1).

as computing resources continue increases, communication resources become the new bottleneck, slowing down system utility growth. Additionally, our proposed scheme outperforms the baseline methods, achieving an improvement of 16.49% to 27.07% in Set ♯2.5 and 17.30% to 27.73% in Set ♯2.6.

C. Latency and Energy Consumption Weight Analysis

Fig. 10(a) illustrates the trade-off between average system latency and energy consumption with respect to the users' latency trade-off weight $\delta^1_{n,m}$, which varies from 0.3 to 0.9. The corresponding energy trade-off weight is $\delta^2_{n,m} = 1-\delta^1_{n,m}$. All results are averaged over 100 independent runs. When $\delta^1_{n,m}$ is low, the algorithm prioritizes energy savings, resulting in lower energy consumption at the cost of higher latency. As $\delta^1_{n,m}$ increases, the algorithm gradually shifts its focus to latency reduction, leading to shorter latency but higher energy usage. This trade-off behavior highlights the flexibility of $\delta^1_{n,m}$ in balancing *low-energy* and *low-latency* objectives to satisfy different application needs. Moreover, Fig. 10(b) presents that the resource utilization ratio remains consistently around 82% across all values of $\delta^1_{n,m}$, indicating that network resources are well-utilized regardless of user preference. The computation time exhibits only minor fluctuations, which stem from the intrinsic randomness of the algorithm rather than from the trade-off weight.



(a) Latency and energy consumption vs. latency weight $\delta^1_{n,m}$. (b) Resource utilization ratio and time consumption vs. latency weight $\delta^1_{n,m}$.

Fig. 10. Latency, energy consumption, resource utilization ratio, and computation time achieved by CNO-TORA under different latency trade-off weights $\delta^1_{n,m}$ in Sets #1.1 (averaged over 100 runs).

VII. Conclusion

In this paper, we investigated an end-edge-cloud collaborative computing network that supports mixed offloading modes (both partial and full offloading), and formulated an optimization problem to maximize users' offloading benefits, which are quantified by a trade-off between task latency reduction and energy consumption savings. To handle the complexity of the formulated problem and enable scalability in large-scale networks, the problem is decomposed and solved in a distributed manner. Extensive simulations based on real-world datasets show that our method outperforms other baselines in terms of overall system utility. The results confirm the effectiveness of the proposed framework in supporting scalable and efficient resource coordination across heterogeneous end-edge-cloud environments. As a result, our method achieves up to 28.15% improvement in offloading gains and maintains 4.01% performance gains even in worst-case scenarios.

Although our method can achieve robust and effective solutions in an end-edge-cloud collaborative computing network, the computation time of Algorithm 1 is relatively high. This is because it requires continuously solving the differential equations (32) and (43). To reduce the computation time of

CNO-TORA, three potential solutions can be considered: (1) implementing the algorithm on specialized hardware [37]; (2) using accelerators to speed up the differential equation solving [45]; and (3) employing learning-based methods for rapid approximation [46]. Future work will focus on integrating these strategies to further enhance computational efficiency while maintaining the high performance of our framework.

References

[1] X. Chen, Y. Zhou, L. Yang, and L. Lv, "User satisfaction oriented resource allocation for fog computing: A mixed-task paradigm," *IEEE Transactions on Communications*, vol. 68, no. 10, pp. 6470–6482, 2020.

[2] Y. Chen, J. Zhao, Y. Wu, J. Huang, and X. Shen, "Multi-user task offloading in uav-assisted leo satellite edge computing: A game-theoretic approach," *IEEE Transactions on Mobile Computing*, 2024, doi: 10.1109/TMC.2024.3465591.

[3] M. Tang and V. W. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Transactions on Mobile Computing*, vol. 21, no. 6, pp. 1985–1997, 2020.

[4] Z. Zhang, N. Wang, H. Wu, C. Tang, and R. Li, "Mr-dro: A fast and efficient task offloading algorithm in heterogeneous edge/cloud computing environments," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3165–3178, 2021.

[5] G. Qu, H. Wu, R. Li, and P. Jiao, "Dmro: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3448–3459, 2021.

[6] D. Wang, H. Zhu, C. Qiu, Y. Zhou, and J. Lu, "Distributed task offloading in cooperative mobile edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 7, pp. 10 487–10 501, 2024.

[7] W. Feng, N. Zhang, S. Li, S. Lin, R. Ning, S. Yang, and Y. Gao, "Latency minimization of reverse offloading in vehicular edge computing," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 5343–5357, 2022.

[8] W. Chen, X. Wei, K. Chi, K. Yu, A. Tolba, S. Mumtaz, and M. Guizani, "Computation time minimized offloading in noma-enabled wireless powered mobile edge computing," *IEEE Transactions on Communications*, 2024.

[9] X. Chen, Y. Cai, L. Li, M. Zhao, B. Champagne, and L. Hanzo, "Energy-efficient resource allocation for latency-sensitive mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2246–2262, 2019.

[10] M. Wu, W. Qi, J. Park, P. Lin, L. Guo, and I. Lee, "Residual energy maximization for wireless powered mobile edge computing systems with mixed-offloading," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 4523–4528, 2022.

[11] Q. Li, M. Guo, Z. Peng, D. Cui, and J. He, "Edge–cloud collaborative computation offloading for mixed traffic," *IEEE Systems Journal*, vol. 17, no. 3, pp. 5023–5034, 2023.

[12] Y. Chen, X. Shen, P. Zhang, S. Lu, L. Wang, Z. Wu, and X. Xie, "Joint optimization of uav-wpt and mixed task offloading strategies with shared mode in sag-piot: A mad4pg approach," *Internet of Things*, vol. 24, p. 100970, 2023.

[13] J. Bi, Z. Wang, H. Yuan, J. Zhang, and M. Zhou, "Cost-minimized computation offloading and user association in hybrid cloud and edge computing," *IEEE Internet of Things Journal*, vol. 11, no. 9, pp. 16 672–16 683, 2024.

[14] M. Kim, J. Jang, Y. Choi, and H. J. Yang, "Distributed task offloading and resource allocation for latency minimization in mobile edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 15 149–15 166, 2024.

[15] R. Huang, W. Wen, Z. Zhou, C. Dong, C. Qiao, Z. Tian, and X. Chen, "Dynamic task offloading for multi-uavs in vehicular edge computing with delay guarantees: A consensus admm-based optimization," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 13 696–13 712, 2024.

[16] Y. Chen, J. Zhao, Y. Wu, J. Huang, and X. S. Shen, "Qoe-aware decentralized task offloading and resource allocation for end-edge-cloud systems: A game-theoretical approach," *IEEE Transactions on Mobile Computing*, 2022.

[17] H. She, L. Yan, and Y. Guo, "Efficient end–edge–cloud task offloading in 6g networks based on multiagent deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 11, no. 11, pp. 20 260–20 270, 2024.

[18] W. Fan, X. Liu, H. Yuan, N. Li, and Y. Liu, "Time-slotted task offloading and resource allocation for cloud-edge-end cooperative computing networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 8, pp. 8225–8241, 2024.

[19] F. Liu, J. Huang, and X. Wang, "Joint task offloading and resource allocation for device-edge-cloud collaboration with subtask dependencies," *IEEE Transactions on Cloud Computing*, vol. 11, no. 3, pp. 3027–3039, 2023.

[20] C.-L. Chen, C. G. Brinton, and V. Aggarwal, "Latency minimization for mobile edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 4, pp. 2233–2247, 2023.

[21] X. Zhu and C. Jiang, "Delay optimization for cooperative multi-tier computing in integrated satellite-terrestrial networks," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 2, pp. 366–380, 2022.

[22] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5031–5044, 2019.

[23] M. Diamanti, C. Pelekis, E. E. Tsiropoulou, and S. Papavassiliou, "Delay minimization for rate-splitting multiple access-based multi-server mec offloading," *IEEE/ACM Transactions on Networking*, 2023.

[24] J. Ren, G. Yu, Y. Cai, and Y. He, "Latency optimization for resource allocation in mobile-edge computation

offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, pp. 5506–5519, 2018.

[25] W. Feng, S. Lin, N. Zhang, G. Wang, B. Ai, and L. Cai, "Joint c-v2x based offloading and resource allocation in multi-tier vehicular edge computing system," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 2, pp. 432–445, 2023.

[26] Z. Niu, H. Liu, J. Du, and Y. Ge, "Partial task offloading for uav-assisted mobile edge computing with energy harvesting," *IEEE Internet of Things Journal*, pp. 1–1, 2024, doi=10.1109/JIOT.2024.3523533.

[27] J. Bi, H. Yuan, K. Zhang, and M. Zhou, "Energy-minimized partial computation offloading for delay-sensitive applications in heterogeneous edge networks," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 4, pp. 1941–1954, 2022.

[28] W. Zhang, Z. Lü, M. Ge, and L. Wang, "Uav-assisted vehicular edge computing system: Min-max fair offloading and position optimization," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 4, pp. 7412–7423, 2024.

[29] T. T. Vu, N. H. Chu, K. T. Phan, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "Energy-based proportional fairness in cooperative edge computing," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 12 229–12 246, 2024.

[30] X. Cao, B. Yang, Y. Shen, C. Yuen, Y. Zhang, Z. Han, H. V. Poor, and L. Hanzo, "Edge-assisted multi-layer offloading optimization of leo satellite-terrestrial integrated networks," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 2, pp. 381–398, 2022.

[31] M. Zhao, R. Zhang, Z. He, and K. Li, "Joint optimization of trajectory, offloading, caching, and migration for uav-assisted mec," *IEEE Transactions on Mobile Computing*, vol. 24, no. 3, pp. 1981–1998, 2025.

[32] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3774–3785, 2020.

[33] H. Yuan, Q. Hu, J. Bi, J. Lü, J. Zhang, and M. Zhou, "Profit-optimized computation offloading with autoencoder-assisted evolution in large-scale mobile-edge computing," *IEEE Internet of Things Journal*, vol. 10, no. 13, pp. 11 896–11 909, 2023.

[34] X. Chen, J. Cao, Y. Sahni, M. Zhang, Z. Liang, and L. Yang, "Mobility-aware dependent task offloading in edge computing: A digital twin-assisted reinforcement learning approach," *IEEE Transactions on Mobile Computing*, vol. 24, no. 4, pp. 2979–2994, 2025.

[35] Q. Liu, H. Zhang, X. Zhang, and D. Yuan, "Joint service caching, communication and computing resource allocation in collaborative mec systems: A drl-based two-timescale approach," *IEEE Transactions on Wireless Communications*, vol. 23, no. 10, pp. 15 493–15 506, 2024.

[36] X. Wang, J. Ye, and J. C. Lui, "Decentralized scheduling and dynamic pricing for edge computing: A mean field game approach," *IEEE/ACM Transactions on Networking*, vol. 31, no. 3, pp. 965–978, 2022.

[37] X. Le, S. Chen, Z. Yan, and J. Xi, "A neurodynamic approach to distributed optimization with globally coupled constraints," *IEEE transactions on cybernetics*, vol. 48, no. 11, pp. 3149–3158, 2017.

[38] Z. Chen, J. Wang, and Q.-L. Han, "A collaborative neurodynamic optimization approach to distributed chiller loading," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[39] G. Li, Z. Yan, and J. Wang, "A one-layer recurrent neural network for constrained nonsmooth invex optimization," *Neural networks*, vol. 50, pp. 79–89, 2014.

[40] X. Li, J. Wang, and S. Kwong, "Hash bit selection based on collaborative neurodynamic optimization," *IEEE Transactions on Cybernetics*, vol. 52, no. 10, pp. 11 144–11 155, 2021.

[41] R. G. Bartle and D. R. Sherbert, *Introduction to real analysis*. John Wiley & Sons, Inc., 2000.

[42] Z. Yan, J. Fan, and J. Wang, "A collective neurodynamic approach to constrained global optimization," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 5, pp. 1206–1215, 2016.

[43] J. Wu, Z. Cao, Y. Zhang, and X. Zhang, "Edge-cloud collaborative computation offloading model based on improved partical swarm optimization in mec," in *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2019, pp. 959–962.

[44] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Proc. Int. Conf. Serv.-Oriented Comput.*, 2018, pp. 230–245.

[45] U. Utkarsh, V. Churavy, Y. Ma, T. Besard, P. Srisuma, T. Gymnich, A. R. Gerlach, A. Edelman, G. Barbastathis, R. D. Braatz *et al.*, "Automated translation and accelerated solving of differential equations on multiple gpu platforms," *Computer Methods in Applied Mechanics and Engineering*, vol. 419, p. 116591, 2024.

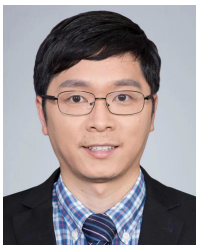
[46] D. Wu and A. Lisser, "Solving constrained pseudoconvex optimization problems with deep learning-based neurodynamic optimization," *Mathematics and Computers in Simulation*, vol. 219, pp. 424–434, 2024.

Changqing Long (Student Member, IEEE) received the B.S. degree in Applied Mathematics and M.S. degree in Operational Research and Cybernetics from the School of Mathematics and Statistics, South-Central Minzu University, Wuhan, China, in 2018 and 2021, respectively. He is currently pursuing the Ph.D. degree in control science and engineering with the College of Control Science and Engineering, Zhejiang University, Hangzhou, China. His current research interests include Edge Computing, Stability Analysis, and Machine Learning.

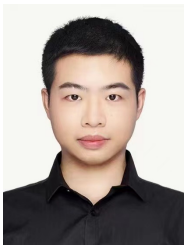
This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2025.3599885

Wenchao Meng (Senior Member, IEEE) received the Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2015. He is currently with the College of Control Science and Engineering, Zhejiang University. His current research interests include adaptive intelligent control, cyber-physical systems, renewable energy systems, and smart grids.

Shizong Li (Student Member, IEEE) received the B.S. degree in electrical engineering and automation from Central South University, Changsha, China, in 2018 and M.S. degree in control science and engineering from Shandong University, Jinan, China, in 2021. He is currently working towards the Ph.D. degree with the School of Control Science and Engineering, Zhejiang University, Hangzhou, China. His research interests include deep learning, edge computing and their application in energy system and industry.



Shibo He (Senior Member, IEEE) received the Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2012. He is currently a Professor with Zhejiang University. He was an Associate Research Scientist from March 2014 to May 2014 and a Post-Doctoral Scholar with Arizona State University, Tempe, AZ, USA, from May 2012 to February 2014. From November 2010 to November 2011, he was a Visiting Scholar with the University of Waterloo, Waterloo, ON, Canada. His research interests include the Internet of Things, crowdsensing, and big data analysis. He serves/served on the Editorial Board for the IEEE Transactions on Network Science and Engineering, IEEE Transactions on Vehicular Technology, Peer-to-Peer Networking and Application (Springer), and KSII Transactions on Internet and Information Systems. He was the General Chair for AIoTsys 2024 and general Co-Chair for iSCI 2022, the Symposium Co-Chair for the IEEE/CIC ICCC 2022, IEEE GLOBECOM 2020, and the IEEE ICC 2017, the TPC Co-Chair for i-Span 2018, the Finance and Registration Chair for ACM MobiHoc 2015, the TPC Co-Chair for the IEEE ScalCom 2014, the TPC Vice Co-Chair for ANT 2013 and 2014, the Track Co-Chair for the Pervasive Algorithms, Protocols, and Networks of EUSPN 2013, the Web Co-Chair for the IEEE MASS 2013, and the Publicity Co-Chair of IEEE WiSARN 2010 and FCN 2014.



Chaojie Gu (Member, IEEE) received the B.Eng. degree in information security from the Harbin Institute of Technology, Weihai, China, in 2016, and the Ph.D. degree in computer science and engineering from Nanyang Technological University, Singapore, in 2020. He was a Research Fellow with Singtel Cognitive and Artificial Intelligence Lab for Enterprise, in 2021. He is currently an Assistant Professor with the College of Control Science and Engineering, Zhejiang University, Hangzhou, China. His research interests include IoT, industrial IoT, edge computing, and low-power wide area network.

Lin Cai (Fellow, IEEE) has been with the Department of Electrical & Computer Engineering at the University of Victoria since 2005, and she is currently a Professor. She is a Royal Society of Canada Fellow, an NSERC E.W.R. Steacie Memorial Fellow, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, and an IEEE Fellow. Her research interests span several areas in communications and networking, with a focus on network protocol and architecture design supporting ubiquitous intelligence. She has been elected to serve the board of the IEEE Vehicular Technology Society (2019-2027), and as its VP in Mobile Radio (2023-2025). She has been a Board Member of IEEE Women in Engineering (2022-2024) and IEEE Communications Society (2024-2026). She has served as an Associate Editor-in-Chief for *IEEE Transactions on Vehicular Technology*, and as a Distinguished Lecturer of the IEEE VTS Society and the IEEE Communications Society.