Journal of Optimization in Soft Computing (JOSC)

Vol. 2, Issue 4, pp: (1-15), Winter-2024
Journal homepage: https://sanad.iau.ir/journal/josc

Paper Type (Research paper)

Real-Time Scalable Task Offloading in Edge Computing Using Semi-Markov Decision Processes and Attention-Based Deep Reinforcement Learning

Abbas Mirzaei[1], Naser Mikaeilvand[2], Babak Nouri-Moghaddam[1], Sajjad Jahanbakhsh Gudakahriz[3], Ailin Khosravani[1], Fatemeh Tahmasebizade[1], Ali Seifi[1], Hosein Hatami[1]

1. Department of Computer Engineering, Ardabil Branch, Islamic Azad University, Ardabil, Iran
2. Department of Computer Engineering, Central Tehran Branch, Islamic Azad University, Tehran, Iran
3. Department of Computer Engineering, Germi Branch, Islamic Azad University, Germi, Iran

Article Info

Article History:
Received: 2024/11/28
Revised: 2025/01/05
Accepted: 2025/02/02

DOI:

Keywords:
Edge Computing; Task Scheduling; Reinforcement Learning; System Scalability.

**Corresponding Author's Email Address: mirzaei_class_87@yahoo.com*

Abstract

Edge computing has emerged as a dynamic framework where computational tasks are offloaded to distributed edge servers (ESs) to provide low-latency and efficient services. As edge systems grow in scale and complexity, leveraging Deep Reinforcement Learning (DRL) has become a prominent approach to optimize task offloading and Resource management. However, traditional DRL-based methodologies encounter several challenges: (1) Discrete-time decision frameworks, such as Markov Decision Processes (MDPs), often enforce offloading in fixed timeslots, leading to scheduling delays and inefficient Resource utilization. (2) Static computational structures struggle to adapt to varying numbers of edge servers or user devices, resulting in scalability issues and system inefficiencies. To overcome these limitations, we introduce a novel DRL-driven real-time offloading mechanism tailored for dynamic and scalable edge environments. Our approach reformulates the offloading problem within a Semi-Markov Decision Process (SMDP) framework and introduces an adaptive optimization mechanism utilizing attention-based graph operations for heterogeneous Resource environments. This system, like how we prioritize tasks and divide resources, figures out how much attention to pay to each task and which server should handle it, to make things work smoothly. To make this work even better in the real world, we use a special method to adjust the rewards, which helps the system learn and improve its performance over time

1. Introduction

The rapid expansion of mobile networks and the proliferation of connected devices have transformed modern computing environments. From autonomous vehicles to immersive augmented reality applications, the demand for high-speed, low-latency services has surged. Traditional cloud computing architectures, despite their powerful centralized Resources, often fall short in meeting these latency-sensitive requirements due to long transmission distances and centralized processing bottlenecks [1]-[5]. This gap has driven the evolution of edge computing, which brings computation and storage closer to end-users by deploying edge servers (ESs) within the network's proximity. Within this paradigm, tasks may be executed locally or offloaded to nearby ESs. While ESs are equipped with more robust computational capabilities compared to UDs, the process of uploading tasks to ESs introduces additional energy consumption and latency. Moreover, the computational capacity of ESs remains constrained compared to centralized

cloud servers, making them unsuitable for handling large volumes of concurrent tasks. Resource contention among multiple tasks can degrade system performance and quality of service (QoS) [6], [7]. Consequently, devising an efficient scheduling mechanism for task offloading has become critical. Such mechanisms aim to optimize the selection of offloading targets and Resource allocation strategies [8], often framed as mixed-integer nonlinear programming (MINLP) problems, which are known to be NP-hard [9].

Initially, mathematical approaches [10] were developed to solve these optimization problems. However, these model-based methods struggle with generalization across diverse edge systems characterized by heterogeneous transmission technologies, application requirements, and computational Resources. To address this limitation, model-free metaheuristic algorithms [11, 12] were introduced for task offloading. Despite their flexibility, these algorithms face significant challenges, including large search spaces and poor adaptability to dynamic edge environments. In recent years, Deep Reinforcement Learning (DRL) has demonstrated exceptional capabilities across various domains, such as robotics control, autonomous driving, and natural language processing. Leveraging deep neural networks, DRL combines high-dimensional data analysis with model-free learning, making it a compelling choice for dynamic edge systems. Its online learning capabilities enable adaptive policy updates through continuous interaction with the environment, offering real-time adaptability to evolving edge conditions. As a result, DRL-based methods have shown promising results in optimizing task offloading and Resource allocation in edge computing [13]-[16]. Despite its advantages, DRL-based approaches face inherent limitations, as illustrated in Fig. 1. Firstly, these methods typically rely on discrete-time Markov Decision Processes (MDPs), where decisions are made at fixed intervals. This framework necessitates batch processing of tasks, causing delays as tasks wait for the next decision interval to be scheduled [17]. Such wait-for-scheduling latency increases Resource contention and lowers task completion rates, particularly in systems with stringent delay requirements. Secondly, traditional DRL methods lack scalability [18, 19]. The fixed computational graph of deep neural networks requires consistent input and output dimensions, making it challenging to adapt to varying system scales [20]. For instance, in mobile edge environments, the dynamic nature of vehicular edge systems—with frequent arrivals and departures of service or user vehicles—renders non-scalable DRL approaches infeasible. Retaining scalability under these conditions is crucial but often necessitates retraining models, a process that is both time-intensive and computationally expensive.

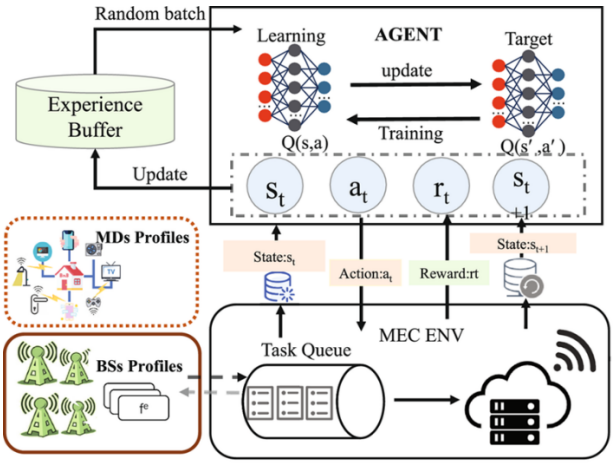

Figure 1. Challenges in DRL-Based Offloading Approaches.

Transitioning from a batched offloading framework to a real-time approach, where tasks are immediately scheduled upon arrival, intuitively minimizes waiting time and avoids dimensional mismatches caused by fluctuating task volumes. However, the discrete-time MDP framework utilized by classical DRL algorithms is inherently unsuitable for such scenarios [21]-[23]. Additionally, scalability challenges, such as mismatches in the dimensions of inputs and outputs caused by dynamic variations in the number of edge servers (ESs) and user devices (UDs), remain unresolved. To address these challenges, we propose a Real-time and Scalable Task Offloading framework (ReSTO), leveraging a DRL-based methodology.

In ReSTO, the task offloading problem is modeled as a Semi-Markov Decision Process (Semi-MDP) to enable decision-making at arbitrary task arrival times. The framework introduces the Scalable Continuous Proximal Policy Optimization (SCPPO) algorithm, specifically designed to align with the

Semi-MDP framework. To ensure scalability, SCPPO employs a heterogeneous graph attention mechanism for feature extraction, translating task-specific characteristics into adaptive attention scores for decision-making. Moreover, we develop a hybrid reward mechanism that integrates model-based and real-time feedback, referred to as the homotopy reward. This reward scheme bridges the

gap between theoretical models and real-world dynamics while enhancing exploration efficiency during learning.

This paper aims to address the limitations of existing DRL-based task offloading approaches in edge computing environments. Specifically, we focus on:

1- Overcoming the limitations of discrete-time MDPs: We propose a novel continuous-time DRL framework that enables real-time, event-triggered task scheduling, eliminating the need for batch processing and reducing wait-for-scheduling latency.

2- Improving scalability in dynamic environments: We introduce a scalable DRL architecture that can The key contributions of this work are as follows:

- **Introduction of ReSTO Framework:**

We propose ReSTO, a novel real-time and scalable task offloading framework. ReSTO models the offloading problem using a Semi-MDP and introduces the SCPPO algorithm for real-time decision-making, eliminating the latency associated with traditional batched scheduling.

- **Scalability via Graph Attention Mechanism:**

SCPPO employs heterogeneous graph attention operations to extract task and Resource features dynamically, enabling adaptive attention score generation. This approach prevents dimensional mismatches as the number of ESs or UDs changes, ensuring scalability.

- **Development of Homotopy Reward:**

We formulate a hybrid reward system combining theoretical model rewards with real-time feedback. This homotopy reward reduces the disparity between theoretical assumptions and real-world conditions, improving both performance and exploration efficiency.

The remainder of this paper is organized as follows: Section II reviews related works, particularly focusing on real-time and scalable RL/DRL-based approaches. Section III presents the system model for real-time offloading and the corresponding optimization problem. In Section IV, we detail the ReSTO framework, including the Semi-MDP formulation and the SCPPO algorithm design. Section V evaluates ReSTO's performance against state-of-the-art algorithms, highlighting its scalability and efficiency. Finally, Section VI concludes the paper with insights and potential future directions.

adapt to varying numbers of tasks and edge servers without requiring extensive model retraining.

By achieving these objectives, we aim to:

- Enhance task completion rates and reduce latency in edge computing systems with stringent performance requirements.
- Improve resource utilization by enabling more efficient task scheduling and allocation.
- Increase the adaptability and robustness of DRL-based offloading solutions in dynamic and unpredictable edge environments.

2. Related Works

In this section, we provide a comprehensive review of DRL-based task offloading methods. Following this, we delve into existing RL/DRL approaches for real-time or scalable task offloading, analyzing their achievements and limitations in comparison to our proposed framework.

A. DRL-Based Task Offloading in Edge Computing

Over the past decade, task offloading in edge computing systems has increasingly relied on Deep Reinforcement Learning (DRL) algorithms due to their capacity for dynamic decision-making and adaptability to complex environments. These algorithms leverage the ability of neural networks to process high-dimensional inputs and learn optimal policies directly through interaction with the environment. Numerous studies have tailored DRL methods to address the unique challenges of edge systems, such as Resource constraints, latency requirements, and dynamic user demands. One notable example is the work of Wang et al. [12], who utilize Deep Q-Learning (DQN) to optimize both task offloading and Resource configuration in a blockchain-enabled edge computing framework. Their approach introduces trust mechanisms and leverages blockchain for secure and efficient offloading. Similarly, Huang et al. [13] employ a Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm for partial offloading systems, where tasks can be split between local and edge processing. This method improves decision-making by accounting for the variability in task size and Resource availability,

demonstrating the potential of DRL in adaptive task allocation.

Building on these foundational approaches, subsequent research has focused on enhancing the performance and robustness of DRL-based task offloading. For instance, Xu et al. [14] and Ma et al. [15] introduce temporal feature extraction to capture the dynamic nature of edge environments, utilizing historical state information to better model system behavior and predict the effects of various actions. This temporal awareness allows the system to adapt to changing workloads and network conditions, leading to more effective offloading strategies.

Moreover, Xu et al. [16] propose an exploration-exploitation strategy tailored to the training process. By prioritizing exploration during the early stages of training and gradually shifting towards the exploitation of learned policies, their approach strikes a balance between discovering new solutions and refining existing ones. This adaptive strategy improves policy performance and ensures more reliable decision-making over time.

To address the computational complexity and convergence challenges associated with large action spaces, researchers have also explored hybrid approaches that integrate DRL with traditional optimization techniques. For example, Chen et al. [17] enhance DQN-based task offloading with sequential quadratic programming for Resource allocation. This combination reduces the dimensionality of the problem and accelerates convergence, enabling more efficient use of edge Resources.

Li et al. [18] take a multi-agent approach, employing a Parameterized Multi-Agent Soft Actor-Critic (SAC) algorithm to address the interdependence of actions across agents. By categorizing actions into those that affect other agents and those that do not, they effectively manage Resource contention in collaborative edge environments. The use of a genetic algorithm further refines Resource allocation decisions, ensuring optimal system performance.

Despite these advancements, existing DRL-based methods face inherent limitations due to their reliance on the discrete-time Markov Decision Process (MDP) framework. This framework enforces decision-making at fixed intervals, leading to batch processing of tasks. Such a structure introduces scheduling delays, as tasks must wait until the next decision point before offloading can occur [24], [25]. This wait-for-scheduling latency becomes particularly problematic in latency-sensitive applications, where even slight delays can significantly degrade performance. Additionally, most DRL approaches encode system states into a one-dimensional input vector for processing by a multi-layer perceptron (MLP). While this design simplifies implementation, it limits scalability. Fixed input-output dimensions in MLPs cannot adapt to changes in the number of edge servers (ESs) or user devices (UDs), resulting in dimensional mismatches. This lack of flexibility hampers the applicability of DRL algorithms in dynamic edge environments, such as vehicular networks or large-scale IoT systems, where the network topology and Resource availability frequently change.

These challenges underscore the need for novel frameworks and algorithms that overcome the constraints of discrete-time MDPs and enable real-time, scalable task offloading in edge computing systems. Future solutions must address both the latency introduced by batch processing and the scalability issues arising from static neural network architectures, paving the way for more adaptive and efficient DRL applications in edge environments.

- Categorization by Objective:
  1. Latency Minimization: Focus on methods specifically designed to minimize task completion time or end-to-end delay.
  2. Energy Efficiency: Analyze methods that prioritize minimizing energy consumption at the device and network levels.
  3. Resource Allocation: Discuss approaches that optimize resource allocation among UDs and ESs, considering factors like CPU, memory, and bandwidth.
  4. Load Balancing: Examine methods that aim to distribute the computational load evenly across the available ESs.

B. Real-Time RL/DRL for Task Scheduling

Real-time decision-making is a critical component of task scheduling in edge computing and numerous other domains, where rapid responses to dynamic changes are essential for maintaining system performance and efficiency. However, the discrete-time Markov Decision Process (MDP) framework, which underpins most traditional RL/DRL methods, introduces inherent constraints when applied to real-time applications. By requiring fixed decision intervals, the discrete-time MDP framework creates bottlenecks, such as

delays in task execution, that compromise the responsiveness and adaptability of RL-based solutions. Alternative frameworks, such as the multi-armed bandit [26]-[30], have been explored to address some of these challenges. While these models are computationally simpler and focus on optimizing immediate rewards, they often fail to account for the temporal dependencies and cumulative effects of actions. This omission can lead to suboptimal decision-making, particularly in complex and dynamic environments where long-term outcomes must be carefully balanced with short-term gains [31]-[33].

In contrast, the Semi-Markov Decision Process (Semi-MDP) framework is particularly well-suited for real-time scheduling tasks. Unlike the discrete-time MDP, Semi-MDP allows for variable intervals between decision points, making it more flexible and capable of handling tasks as they arrive. This flexibility enables the development of policies that optimize long-term performance while addressing the immediate requirements of real-time systems. For instance, Liang et al. [20] and Hao et al. [21] successfully use Semi-MDPs to model real-time scheduling problems, demonstrating the framework's potential to accommodate dynamic workloads and varying system conditions. Despite its advantages, adapting existing algorithms to the Semi-MDP framework poses unique challenges due to its structural differences from the traditional MDP approach. One common strategy involves normalization, which converts Semi-MDP problems into an MDP-compatible format, allowing established DRL algorithms to be applied. For example, Liang et al. [22] normalize Semi-MDP problems by estimating theoretical model-based Q-values for supervised pre-training [34]-[36]. This approach provides a starting point for the policy, which is then refined through interactions with the environment. Similarly, Wu et al. [23] utilize state transition probabilities during the normalization process to transform Semi-MDPs into a form solvable by value iteration techniques.

An alternative to normalization-based methods is the direct design of algorithms tailored to the Semi-MDP framework. These approaches avoid the approximations and assumptions inherent in normalization, enabling more accurate modeling of real-world scenarios. For example, Van Huynh et al. [24] propose a Dueling Double Deep Q-Network (DDQN) approach that maximizes cumulative single rewards without incorporating discount factors, focusing instead on immediate benefits within a Semi-MDP structure. Wei et al. [9] employ an exponential decay model to compute cumulative discounted returns, deriving a Bellman optimality equation to guide decision-making with DQN. Kim et al. [25] adapt the Soft Actor-Critic (SAC) algorithm for the Semi-MDP framework, introducing modifications that account for the variable time intervals and cumulative reward structures characteristic of Semi-MDPs. Despite these advancements, existing methods still exhibit notable limitations. Normalization-based approaches often rely heavily on theoretical assumptions, such as idealized transition models or fixed state representations, which reduce their generalizability to real-world, complex environments [37]-[40]. These assumptions can lead to performance degradation when applied to heterogeneous and highly dynamic edge systems, where practical constraints and unpredictable factors frequently deviate from theoretical models. On the other hand, model-free DRL approaches [41]-[45] that bypass theoretical dependencies also face challenges. These methods commonly employ simplistic neural network architectures, such as basic feedforward models, that lack the scalability needed to adapt to dynamic edge network conditions. In systems where the number of edge servers (ESs) and user devices (UDs) can fluctuate significantly, fixed input-output dimensions lead to dimensional mismatches, requiring costly retraining of the models to accommodate changes [46]-[48]. This inflexibility limits the practical deployment of model-free DRL solutions in scenarios characterized by high variability and evolving system requirements. Overall, while the Semi-MDP framework offers significant potential for enabling real-time decision-making in edge computing, achieving effective and scalable solutions necessitates innovative algorithmic designs that address both the limitations of normalization-based methods and the scalability constraints of traditional DRL models. Future work must focus on bridging these gaps to develop robust and adaptable frameworks capable of supporting real-time, scalable task scheduling in edge environments.

- Weaknesses of Current Semi-MDP Methods:
  1. Normalization-Based Approaches:
  2. Reliance on Theoretical Assumptions: Often rely on idealized models and assumptions, which can limit their applicability in real-world scenarios with high variability and uncertainty.
  3. Potential for Accuracy Loss: The normalization process can introduce approximations that may lead to

suboptimal solutions or reduced accuracy.

4. Limited Exploration of Direct Semi-MDP Algorithms: While some direct approaches exist, the field is still relatively under-explored compared to normalization-based methods.
5. Scalability Challenges: As the complexity of the environment and the number of tasks increase, solving Semi-MDPs can become computationally expensive, especially for complex DRL algorithms.
6. Handling of Uncertainty: Many existing methods may not adequately address the inherent uncertainty and stochasticity present in real-world scheduling problems.

3. System Model and Problem Formulation

We consider a crowdsourcing-inspired MEC system, as illustrated in Fig. 1, comprising multiple applications and edge servers (ESs) with diverse configurations and characteristics. These applications may vary significantly in their requirements, encompassing delay-sensitive services such as networked gaming, autonomous driving, and AR/VR, as well as resource-intensive tasks like big data analytics, scientific computing, and video surveillance [49]. Similarly, ESs can range from micro data centers and edge clouds to high-capacity computing servers or even gateways deployed in residential or office settings. For generality, we assume these ESs are managed and operated by distinct edge service providers. To maximize resource utilization and enhance system performance in terms of scalability, reliability, and other metrics, a third-party platform is introduced to coordinate ES operations and handle workload dispatch from end users. Acting as an intermediary, this platform serves as a front-end interface for edge computing services, bridging the gap between clients submitting tasks and ESs providing computational resources. Upon receiving a task, the platform assigns it to the most suitable ES hosting the requested service and ensures the computation result is returned to the client seamlessly. This interaction is transparent to users, provided the system meets their application performance expectations, such as low latency and high computation quality.

Both application providers and ESs must undergo an onboarding process with the platform before accessing or delivering edge services. This formalized process involves signing agreements with the platform to define roles and responsibilities. For application providers, this includes specifying service requirements such as task rates, task valuation, budget constraints, computational demands, QoS parameters (e.g., maximum tolerable delay), and security or compliance needs. Similarly, ESs seeking to participate in the system are subject to a comprehensive evaluation by the platform. This involves reviewing their security protocols, compliance certifications, and data management practices to ensure adherence to industry standards and regulatory requirements [50]. Additionally, a risk assessment is often conducted to identify potential vulnerabilities. ESs must provide detailed information regarding their resource capacities, operational costs, and revenue expectations.

Using this information, the platform optimizes task offloading strategies and resource allocation for ESs, subsequently formalizing agreements with both parties. Once agreements are in place, ESs configure the necessary accounts and infrastructure, enabling application providers to deploy their services. Importantly, ongoing monitoring and auditing mechanisms are established to ensure all parties adhere to the agreed-upon terms, with regular performance and compliance evaluations conducted throughout the service lifecycle.

This study considers a scenario where application providers make advance payments to the platform, which, in turn, allocates a portion of these payments to incentivize contributions from edge servers (ESs). The platform's key decisions include: (1) whether to accept both the application providers and ESs into the system, (2) determining the amount of resources each ES should allocate to applications, and (3) devising an efficient task dispatching strategy to distribute tasks among the backend ESs hosting the services. To simplify notation, we define the set of ESs and applications/services in the system as M and N, respectively, with the corresponding cardinalities denoted by |M| and |N|. For clarity, the terms "applications" and "application providers" are used interchangeably in this paper unless otherwise specified. The primary notations employed throughout this work are summarized in Table 1.

Each application $i \in N_i$ is characterized by a tuple $(p_i, v_i, \alpha_i, D_i, s_i)$,where:

1. p_i: The payment made by application provider iii to the platform for task offloading.

2. v_i: The utility gained by i from offloading a task, such as reduced energy consumption at user devices, enhanced computational quality, or shorter response times. Generally, $p_i \leq v_i$ Offloading offers net benefits to the application.
3. α_i: The arrival rate of tasks for application i.
4. s_i: The workload (measured in CPU cycles) required to process a task.
5. D_i: The maximum latency tolerable by application i.

Given the stochastic nature of the system and the uncertainty in resource allocation at ESs, the actual value derived by an application from task offloading depends on the quality of the edge computing service. We represent this with a utility function $u_{ij} \in [0,1]$, which quantifies the satisfaction level of application i when offloading tasks to ES_j. This utility function is an abstract representation and can vary depending on the application's requirements.
For instance, for delay-sensitive applications, u_{ij}may be defined based on reductions in task latency. For resource-intensive applications, u_{ij}could reflect the computational quality, such as compression ratios or prediction accuracy. Moreover, the form of u_{ij} can differ even within the same application category. For example, in delay-sensitive applications, u_{ij} could be a step function to model satisfaction levels in the presence of hard deadlines.

$$u_{ij} = \begin{cases} 1, & \text{if } t_{ij} \leq D_i \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

A. Platform Model

The platform operates under the following assumptions:

1. The platform employs a probabilistic task dispatching mechanism, where each application task is routed to a specific ES based on predefined probabilities.
2. The payment p_i made by application iii is distributed between the platform and the ES executing the task. Specifically, the ES receives a reward of $(1-\lambda_i)p_i$, where $\lambda_i \leq 1$, while the platform retains $\lambda_i p_i$ as its service charge or maintenance fee. The parameter λ_i , a critical system variable, is determined by the platform and forms part of the contractual agreement with the ES.

4. Real-time and Scalable Task Offloading Framework

Before detailing the algorithm, we first describe the calculation of u_{ij} and t_{ij} under a fixed resource allocation $F_{ij} = F_0$. The following assumptions, drawn from prior studies, are applied:

1. Tasks from each application arrive according to a Poisson process [46]. Consequently, the arrival of tasks from application i at ES_j also follows a Poisson process with a rate of $r_{ij} = \alpha_i x_{ij}$, where α_i represents the task arrival rate, and x_{ij} denotes the probability of task dispatch to ES_j .

2. The workload of tasks from each application is assumed to follow an exponential distribution (in CPU cycles) [27][36]. This implies that the processing time for a task from application i at ES_j also follows an exponential distribution with a mean of 1/wij1/w where $w_{ij} = F_{ij}(0)$ and s_i represents the workload of the task.

Based on these assumptions, the task processing system for an application i at ES_j can be modeled as an M/M/1queue. The probability density function (pdf) for the task delay t_{ij} this system is then expressed as:

$$f_T(t_{ij} \leq t) = (w_{ij} - r_{ij}) \cdot e^{-(w_{ij}-r_{ij})t} \quad (2)$$

Assuming u_{ij} is defined as in Eq. (2) and $x_{ij} > 0$ (indicating that tasks from application i are offloaded to ES_j), the relationship derived from constraint (3b) is as follows:

$$\Pr\left(t_{ij} < \left(1 - \frac{p_i}{v_i}\right) D_i\right) \geq \text{prob}_i \quad (3)$$

Combining (7) and (8), we get:

$$x_{ij} \leq \frac{1}{\alpha_i}\left[\frac{\ln(1-prob_i)}{\left(1-\frac{p_i}{v_i}\right)D_i} + \frac{F_{ij}^0}{s_i}\right] \quad (4)$$

Let $x_{ij}H_x$ denote the right-hand side (RHS) of the inequality mentioned above, defined as:

$$x_{ij}^{H} \triangleq \frac{1}{\alpha_i}\left[\frac{\ln(1-prob_i)}{\left(1-\frac{p_i}{v_i}\right)D_i}+\frac{F_{ij}^{0}}{s_i}\right] \quad (5)$$

Clearly, $x_{ij}H_x$ represents the upper bound of the offloading probability for which application provider i is satisfied with offloading its tasks to ES_j, meeting the QoS requirements. Notably, this upper bound is independent of λ_i and is solely determined by F_{ij} (0) and the workload profiles.

Similarly, from constraint (3c) and assuming $x_{ij} > 0$, we derive:

$$x_{ij} \geq \frac{(1+\beta_{ji})c_j\left(F_{ij}^{0}\right)}{\alpha_i(1-\lambda_i)p_i}, \quad (6)$$

Let the right-hand side (RHS) of the above inequality be denoted as x_{ij}^{L}, defined as:

$$x_{ij}^{L} \triangleq \frac{(1+\beta_{ji})c_j\left(F_{ij}^{0}\right)}{\alpha_i(1-\lambda_i)p_i}. \quad (7)$$

Algorithm 1 Deriving the optimal resource allocation, task offloading probabilities, and ratios under a given resource allocation $F_{ij}^{0\prime}$s.

Input: Task profiles $(\alpha_i's, p_i's, v_i's, D_i's, s_i's)$; ES profiles $\left(c_j(F_{ij})'s, F_j's, \beta_{ji}'s\right)$; Initial resource allocations $F_{ij}^{0\prime}s$;

Output: Resource allocations $F_{ij}^{0,\sim\prime}s$; Ratios $\lambda_i^{0,\sim\prime}s$; Task offloading probabilities $x_{ij}^{0,\sim\prime}s$;

for $i \in \mathcal{N}$ ***do***
  for $j \in \mathcal{M}$ ***do***
    Derive x_{ij}^{H} and λ_{ij}^{0} according to Eq 5.
for $i \in \mathcal{N}$ ***do***
  Get λ_i^{0} and x_{ij}^{0}
for $j \in \mathcal{M}$ ***do***
  Get $\mathcal{Y}_{ij}^{0}$ ś
Obtain $\lambda_i^{0,\sim}$ ś, $F_{ij}^{0,\sim}$ś and $x_{ij}^{0,\sim}$ś

5. Simulation Experiments

A. Experimental Setup

The simulation framework was developed using Python 3.9 and PyTorch 2.3.0, running on a high-performance desktop system powered by an Intel Core i9-13900K processor and an Nvidia GeForce RTX 3090 GPU. This computational setup was chosen to ensure efficient processing of the complex algorithms and large-scale datasets involved. The simulation leverages vehicle trajectory data from the Peachtree Street section of the Next Generation Simulation (NGSIM) dataset [36]. This dataset provides detailed and realistic representations of urban traffic flow, making it suitable for modeling dynamic user-device behaviors in edge computing scenarios.

In our simulation environment, user devices (UDs) are designed to move along stochastic trajectories generated from the NGSIM dataset. These trajectories simulate real-world mobility patterns, such as vehicles traveling through a busy metropolitan area. UDs are assumed to exit the system once their respective trajectories conclude, reflecting the dynamic entry and exit behavior typical in edge networks. Edge nodes (ENs) are deployed strategically at random locations along these trajectories, ensuring adequate coverage of user mobility patterns while capturing the inherent randomness of real-world deployments. The system parameters used in the simulation are comprehensively detailed in Table II. These include network characteristics, Resource configurations, and mobility patterns, ensuring that the simulation accurately reflects the operational constraints and requirements of modern edge computing environments.

Training Process and Network Design

The training process was meticulously designed to optimize the learning performance of the proposed algorithm. The neural network architecture incorporates several specialized components to handle the complexity of real-time task offloading and Resource allocation. The hidden feature dimension d was set to 256, balancing computational efficiency with model expressiveness. The attention mechanism employed K=4 attention heads, enabling the model to capture intricate relationships between tasks and edge nodes across multiple dimensions.

Three encoder components—H_{EN}, H_{Cell}, and H_{Task} —were implemented as two-layer multilayer perceptrons (MLPs), each employing Tanh activation functions. These encoders transform raw input data into high-dimensional representations suitable for downstream processing. The MLP Dc, responsible for computing Resource allocation, was configured with two layers, ensuring lightweight and efficient computation. In contrast, the MLP Dv within the critic network was designed with four layers to enhance its capacity for estimating value functions,

which are critical for effective policy evaluation and improvement.

Key Parameters for the Continuous-Time PPO Algorithm

To align the training process with the Semi-Markov Decision Process (Semi-MDP) framework, we tailored the continuous-time Proximal Policy Optimization (PPO) algorithm with carefully selected hyperparameters. The discount factor α was set to 0.1, ensuring a balanced emphasis on immediate rewards and long-term gains. The importance sampling ratio ϵ, set to 0.2, controlled the degree of policy updates to maintain stability during training. The Generalized Advantage Estimation (GAE) hyperparameter λ was configured as 0.98 to improve the estimation of advantages, enhancing the convergence rate and overall learning efficiency.

B. Training Configuration and Iterations

The training process spanned 400 episodes, providing sufficient iterations for the algorithm to converge to an optimal policy. Each episode was further divided into a maximum of 200 iterations, allowing the model to explore diverse states and actions comprehensively. During training, the model continually interacted with the simulated environment, refining its policy through trial and error while leveraging feedback from the homotopy reward mechanism. This hybrid reward system combined theoretical insights with real-time observations, bridging the gap between simulated models and practical deployments.

The overall design of the simulation environment, coupled with the robust training setup, ensures that the proposed algorithm is well-equipped to handle dynamic and scalable edge computing scenarios. By incorporating realistic mobility patterns, stochastic task generation, and advanced neural network architectures, the simulation framework provides a reliable foundation for evaluating the effectiveness of real-time task offloading and Resource allocation strategies in next-generation edge systems.

TABLE I.
PARAMETER SETTINGS OF SIMULATION

Notations	Simulation Value	Notations	Simulation Value
M	8	α	U(1.0, 1.2)
$\boldsymbol{f_m}$	U (2, 4) GHz	β	MB U (0.8, 1.0) GCycle
$\boldsymbol{d^m}$	50 Meter	ϑ	U (1, 2) Second
N	30	p	
			1 Watt
$\boldsymbol{q^{max}}$	3	ς	-3
$\boldsymbol{\iota}$	1	σ^2	-114 dBm/MHz
X	0.1	B	1 MHz
$\boldsymbol{f_n}$	U (1, 2) GHz	Ω	1
ϒ	4	K	10^{-27}

The data reuse frequency was configured to 10 iterations. For the actor-network, the learning rate was set to 1×10^{-4}, while the critic network utilized a higher learning rate of 1×10^{-3}.The Adam optimizer, with $\varepsilon = 1 \times 10^{-5}$,was employed for parameter updates.

To evaluate the performance of the proposed method, we conducted a comparative analysis with four state-of-the-art DRL-based methods designed to address scalability, as well as a single-step greedy method. A brief overview of these approaches is as follows:

- **Single-Step Greedy (SSG):** This method selects actions greedily based on immediate task benefits. While intuitive, it focuses exclusively on short-term gains, neglecting long-term system optimization.
- **Sequence to Sequence (S2S) [11]:** This approach leverages recurrent neural networks (RNNs) for sequential system feature extraction and multi-action generation. However, it operates under a batched offloading framework and struggles to adapt action dimensions to dynamic variations in the number of edge nodes (ENs).
- **Self-Attention (SA) [10]:** Using a self-attention mechanism, this method integrates task features and generates actions in parallel. Despite this, it inherits the limitations of S2S, including reliance on batched offloading and the inability to adapt to changes in EN counts due to its concatenation of EN states as input.
- **Event-Driven DQN (EDQ) [9]:** This real-time approach employs an event-driven Deep Q-learning framework based on task and EN states. However, its reliance on a multilayer perceptron (MLP) architecture for the Q-network constrains scalability, particularly in large-scale systems.
- **GNN-based Multi-agent DRL (GMD) [30]:** This method utilizes a distributed multi-agent DRL framework with graph neural

networks (GNNs), allowing user devices (UDs) to independently select actions. By representing offloading targets as positive integers instead of one-hot vectors, it offers significant scalability. However, multi-agent DRL frameworks are challenging to train in large-scale environments, often leading to diminished performance.

For a fair comparison, we set the batch interval to 0.2 in subsequent experiments for the S2S, SA, and GMD methods, which follow a batched offloading framework.

Notably, the ReSTO framework outperformed all baselines in terms of system cost, even under zero-shot transfer scenarios, surpassing re-trained methods as well. This underscores the exceptional scalability and efficiency of ReSTO. Interestingly, we observed that the system costs of SA and EDQ remained stable or even increased as additional ENs became available. This phenomenon is attributable to their reliance on concatenated EN states as input, which inflates the input dimensions, causing the critic network to struggle with accurate evaluations. For EDQ, the increase in selectable actions further complicates Q-network convergence, exacerbating its limitations in larger systems.

C. Batched Offloading V.S. Real-Time Offloading

To highlight the performance benefits of transitioning from batched offloading to real-time offloading, we compare the proposed ReSTO method with existing approaches under two load scenarios. The results, as illustrated in Fig. 2, consider a normal scenario with baseline system settings and a harsh scenario where the load factor $\beta \in u(1.2,1.4)$. For consistency, we introduce artificial delays in task execution to emulate batched offloading for ReSTO, SSG, and EDQ, which inherently support real-time offloading. Other methods, lacking real-time capabilities, are excluded from this analysis. Batched offloading is tested with four discrete timeslot intervals: 0.8, 0.6, 0.4, and 0.2.

The experimental findings indicate that reducing the interval duration in batched offloading substantially lowers system costs under both load scenarios, with the real-time offloading approach consistently achieving the best performance. This improvement is especially pronounced under higher load conditions, as shorter decision intervals minimize the delay between task arrival and scheduling, allowing for more effective Resource management. Conversely, under increased system loads, extended waiting periods in batched offloading sharply reduce the scope for scheduling adjustments, leading to greater performance degradation. Notably, at elevated load levels with larger timeslot intervals, DRL-based methods display inferior performance compared to the SSG approach. This can be attributed to challenges in learning from delayed and sparse rewards during training, particularly when task failures dominate the early learning phase. As a result, many DRL-based methods converge to suboptimal solutions, unable to recover effectively. In contrast, the ReSTO framework, supported by the homotopy reward mechanism, provides more immediate and structured reward feedback during early training stages. This design facilitates more efficient exploration and allows ReSTO to avoid local optima, delivering significantly better performance even under harsh conditions.

D. Ablation Study

An ablation study was conducted to investigate the impact of the homotopy reward design and graph-based cell state aggregation on the performance of the proposed framework. The experiments were carried out under both normal and harsh scenarios to provide a comprehensive evaluation across varying load levels. Two key components were evaluated: (1) the reward mechanism, with three configurations considered—model-based reward, reality reward, and the proposed homotopy reward—and (2) the user device (UD) state fusion method, comparing direct aggregation of UD states independently versus graph-based aggregation of cell states. These configurations were systematically combined into multiple algorithm variants, and their performance was assessed.

The study revealed significant differences in performance across the reward settings. Among the configurations, the reality reward (blue line) exhibited the largest fluctuations during training. These fluctuations can be attributed to the reward mechanism's reliance on real-time feedback, which is inherently noisy and less predictable. The lack of robust guidance in the early training stages often led to instability in task success rates, particularly under harsh scenarios where Resource constraints are more pronounced. Additionally, this configuration struggled to balance immediate performance with long-term optimization, highlighting its limitations in dynamic and unpredictable environments.

Conversely, the model-based reward demonstrated greater stability but was less effective in capturing the complexities of real-world conditions. This

resulted in suboptimal exploration, limiting its ability to adapt to diverse scenarios. The proposed homotopy reward bridged the gap between the model-based and reality rewards, effectively integrating theoretical guidance with real-time feedback. This hybrid approach significantly improved exploration efficiency, enabling the algorithm to converge faster and achieve better performance across both normal and harsh scenarios. The homotopy reward design also mitigated the challenges of sparse rewards, ensuring consistent progress during training.

The study further examined the effects of state aggregation methods. Directly aggregating UD states independently often resulted in subpar system performance due to the lack of contextual understanding of Resource and task interactions within the network. In contrast, the graph-based cell state aggregation effectively captured spatial and temporal dependencies, enhancing the framework's ability to adapt to changes in system dynamics. By leveraging graph structures to model interactions between tasks and edge servers (ESs), this method provided a holistic view of the network, leading to more informed and efficient decision-making.

The analysis also sheds light on the limitations of the GMD algorithm, which demonstrated a tendency to prioritize tasks with higher energy consumption. This behavior can be traced to its distributed multi-agent DRL framework, where each agent operates with limited visibility into the overall system state. Without a comprehensive view of the network, agents often opted to process tasks at a higher frequency to minimize CPU occupancy and avoid Resource contention. While this strategy may reduce immediate delays, it inadvertently increases energy consumption and diminishes the overall system efficiency.

In summary, the results highlight the advantages of the proposed homotopy reward design and graph-based cell state aggregation in improving system performance and scalability. By addressing the shortcomings of traditional reward mechanisms and state aggregation methods, the proposed approach achieves superior stability, faster convergence, and enhanced adaptability, particularly under challenging operational conditions.

E. Comparisons under Different Environmental Settings

This section evaluates the performance of our proposed algorithm against other methods under varying simulation parameters, specifically focusing on the task generation interval parameter (Ω) of the exponential distribution and the user preference for required CPU cycles $\overline{\beta}$). These parameters influence the system load by altering the task arrival rate and the computational demand of each task. Our analytics illustrate the system costs across different values of Ω. A reduction in Ω corresponds to an increased number of tasks and a heavier overall system load. The results reveal that DRL-based methods consistently outperform the SSG approach in all scenarios. This is due to the long-term optimization capabilities inherent in DRL, which enable proactive and foresight-driven decision-making. In contrast, the SSG method prioritizes immediate task optimization without accounting for future system demands, leading to significant queue delays and higher overall costs.

Among the DRL-based methods, the S2S approach exhibits comparatively higher system costs. This can be attributed to its vulnerability to the memory-forgetting issue associated with processing long task sequences. As Ω decreases, the number of tasks requiring scheduling within each discrete timeslot increases, further amplifying this limitation. In contrast, the proposed ReSTO framework achieves the lowest system cost across all scenarios, with the performance gap widenthe ing as Ω decreases. This superior performance stems from the fundamental differences between real-time and batched offloading. As the system load intensifies with a higher task arrival rate, the limitations of batched offloading become more pronounced, leading to greater performance degradation for methods relying on discrete scheduling intervals. These findings reaffirm the advantages of the real-time offloading strategy employed in ReSTO, particularly under high-load conditions.

Our analytics compares the performance of the algorithms across different values of $\overline{\beta}$, which represents the computational load associated with tasks. Higher $\overline{\beta}$ values indicate that tasks demand more CPU cycles for processing, thereby increasing the system load. The results reveal that under low-load scenarios, DRL-based methods demonstrate a clear advantage over the Single-Step Greedy (SSG) approach, achieving significantly lower system costs. This improvement is attributed to the long-term optimization capabilities of DRL, which enable more efficient Resource allocation and task scheduling by anticipating future system states. In contrast, SSG focuses solely on immediate task optimization, often resulting in suboptimal Resource utilization and increased queuing delays. As the system load intensifies with higher $\overline{\beta}$ values, the performance gap between

DRL-based methods and SSG narrows. This reduction in effectiveness stems from the challenges introduced by the more demanding environment. Heavier system loads generate delayed and sparse rewards, complicating the training process for DRL algorithms and limiting their ability to converge to optimal policies. Under these conditions, traditional DRL-based approaches are more likely to become trapped in local optima, as the sparse feedback makes it difficult to identify and reinforce effective scheduling strategies.

The proposed ReSTO framework, however, addresses these limitations through its innovative homotopy reward mechanism. By combining model-based and reality-based rewards, the homotopy reward provides consistent and structured feedback throughout the training process. This design enables ReSTO to navigate complex and dynamic system states more effectively, avoiding local optima and guiding the algorithm toward globally optimized solutions. The ability to adapt to varying load conditions is further enhanced by the real-time offloading strategy employed in ReSTO, which eliminates the delays associated with batched scheduling. This combination of timely decision-making and robust reward feedback allows ReSTO to maintain superior performance across all load conditions. Moreover, the advantages of ReSTO become increasingly pronounced as the system load rises. In high-load scenarios, where tasks require significant computational Resources and delays are more detrimental, the benefits of real-time offloading are particularly evident. By reducing the waiting time between task arrival and execution, ReSTO not only minimizes queuing delays but also maximizes Resource utilization efficiency. These factors collectively contribute to ReSTO's consistent outperformance of competing methods, demonstrating its scalability, adaptability, and resilience under diverse operational conditions.

In summary, the integration of the homotopy reward mechanism and real-time offloading in ReSTO provides a significant edge over existing DRL-based approaches and heuristic methods like SSG. The framework's ability to maintain low system costs under both low and high system loads highlights its robustness and makes it a promising solution for real-time and scalable task offloading in dynamic edge computing environments.

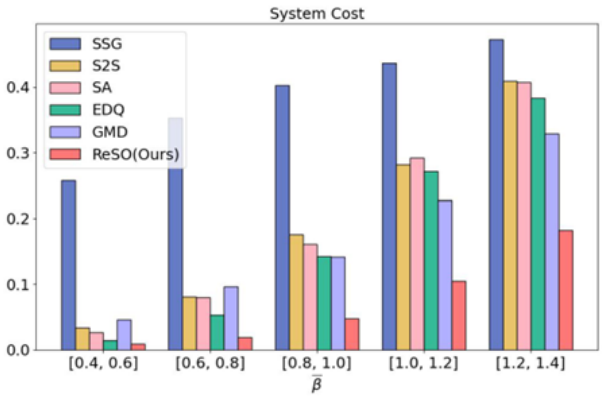

Fig 2. System Costs Across Algorithms for Varying Task CPU Cycle Requirements.

6. Conclusions

While DRL-based algorithms have demonstrated exceptional capabilities in optimizing task offloading for edge computing, several persistent challenges limit their potential for broader practical deployment. Key among these is the waiting time associated with batched decision-making and the dimensional mismatches arising from dynamic system scales. These limitations not only impede performance improvements but also hinder the scalability and adaptability of such methods in real-world applications. To address these critical issues, we introduce ReSTO, a DRL-driven real-time and scalable offloading framework designed to overcome the inherent challenges of existing methods. ReSTO redefines the task-offloading paradigm by shifting from a batched scheduling approach to a real-time offloading framework. Tasks are scheduled immediately upon arrival, eliminating waiting times and enabling more efficient Resource utilization. This is achieved by modeling the offloading problem as a Semi-Markov Decision Process (Semi-MDP), allowing decision-making at arbitrary task arrival times rather than fixed intervals. To effectively solve the problem, ReSTO employs a novel continuous-time Proximal Policy Optimization (PPO) algorithm, enhanced with specially designed scalable actor and critic networks that adapt seamlessly to varying numbers of edge nodes (ENs) and user devices (UDs). This architecture ensures robust performance across dynamic system conditions.

In addition to its innovative decision-making framework, ReSTO introduces two key mechanisms to further enhance its performance. First, the homotopy reward mechanism integrates model-based and reality-based rewards to bridge the gap between theoretical assumptions and real-world dynamics. This approach improves learning efficiency, enabling the algorithm to avoid local optima and converge toward globally optimal policies. Second, ReSTO clusters UDs into cells,

aggregating state information to reduce dimensional complexity and improve decision accuracy. This clustering approach ensures scalability and effective Resource allocation even in large-scale systems with high task loads. Extensive experimental evaluations highlight the significant advantages of ReSTO over state-of-the-art algorithms. The results demonstrate that ReSTO consistently achieves lower system costs while exhibiting better scalability as the number of ENs and UDs fluctuates. These findings underscore the robustness and adaptability of the proposed framework, making it well-suited for the dynamic and heterogeneous environments characteristic of modern edge computing systems. However, transitioning from batch to real-time offloading also brings new challenges, particularly in terms of the computational overhead associated with state acquisition and decision-making processes. The need for rapid, real-time decisions places greater importance on minimizing time complexity to ensure the practical viability of ReSTO in large-scale deployments. Future work will focus on exploring and developing algorithms with reduced time complexity, capable of operating under partially updated or approximate state information. By addressing these challenges, we aim to further enhance the efficiency and scalability of real-time offloading solutions, paving the way for their widespread adoption in edge computing.

- Experimental Results and Validation:

Extensive simulations demonstrate the superior performance of ReSTO compared to state-of-the-art methods. Specifically, ReSTO consistently achieves lower system costs (e.g., energy consumption, latency) while exhibiting better scalability as the number of ENs and UDs fluctuates. These results validate the effectiveness of ReSTO in optimizing resource allocation and adapting to dynamic system conditions.

Conceptual Explanations:

- Addressing Batching Limitations: By moving to a real-time framework, ReSTO eliminates the inherent delay associated with batched decision-making, leading to more responsive and efficient resource allocation.

References

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A survey on mobile edge computing: The communication perspective,” IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2322–2358, 2017.

[2] Mirzaei, A. and Najafi Souha, A., 2021. Towards optimal configuration in MEC Neural networks: deep learning-based optimal resource allocation. Wireless Personal Communications, 121(1), pp.221-243.

[3] Zhou, Guoliang, and Amin Mohajer. "Blind reconfigurable intelligent surfaces for dynamic offloading in fixed-NOMA mobile edge networks." International Journal of Sensor Networks 46, no. 3 (2024): 142-160.

[4] H. Guo, J. Li, J. Liu, N. Tian, and N. Kato, “A survey on space-airground- sea integrated network security in 6g,” IEEE Communications Surveys & Tutorials, vol. 24, no. 1, pp. 53–87, 2022.

[5] Duan, H., & Mirzaei, A. (2023). Adaptive Rate Maximization and Hierarchical Resource Management for Underlay Spectrum Sharing NOMA HetNets with Hybrid Power Supplies. Mobile Networks and Applications, 1-17.

[6] Zhou, Nan, Ya Nan Li, and Amin Mohajer. "Distributed capacity optimisation and resource allocation in heterogeneous mobile networks using advanced serverless connectivity strategies." International Journal of Sensor Networks 45, no. 3 (2024): 127-147.

[7] X. Huang, Y. Chen, J. Liu, M. Wang, P. Li, and Q. Zhao, “Joint interdependent task scheduling and energy balancing for multi-uav enabled aerial edge computing: A multi-objective optimization approach,” IEEE Internet of Things Journal, vol. 10, no. 4, pp. 3147–3160, 2023.

[8] Z. Yang, C. Pan, K. Wang, and M. Shikh-Bahaei, “Energy efficient Resource allocation in uav enabled mobile edge computing networks,”IEEE Transactions on Wireless Communications, vol. 18, no. 9, pp. 4576–4589, 2019.

[9] Mohajer, Amin, Mohammad Yousefvand, Ehsan Noori Ghalenoo, Parviz Mirzaei, and Ali Zamani. "Novel approach to sub-graph selection over coded wireless networks with QoS constraints." IETE Journal of Research 60, no. 3 (2014): 203-210.

[10] X. Zhang, J. Zhang, J. Xiong, L. Zhou, J. Wei, and H. Li, “Energyefficient multi-uav-enabled multiaccess edge computing incorporating noma,” IEEE Internet of Things Journal, vol. 7, no. 6, pp. 5613–5627, 2020.

[11] Mirzaei, A. (2022). A novel approach to QoS-aware resource allocation in NOMA cellular HetNets using multi-layer optimization. Concurrency and Computation: Practice and Experience, 34(21), e7068.

[12] T. Zhang, Y. Xu, J. Loo, D. Yang, L. Xiao, and Y. Zhao, “Joint computation and communication design for uav-assisted mobile edge computing in iot,” IEEE Transactions on Industrial Informatics, vol. 16, no. 8, pp. 5505–5516, 2020.

[13] Z. Liu, X. Tan, M. Wen, S. Wang, C. Liang, and Q. Zhao, “An energyefficient selection mechanism of relay and edge computing in uavassisted cellular networks,” IEEE Transactions on Green Communications and Networking, vol. 5, no. 3, pp. 1306–1318, 2021.

[14] Mohajer, Amin, Javad Hajipour, and Victor CM Leung. "Dynamic Offloading in Mobile Edge Computing with Traffic-Aware Network Slicing and Adaptive TD3 Strategy." IEEE Communications Letters (2024).
[15] Yang, Jiuting, and Amin Mohajer. "Multi objective constellation optimization and dynamic link utilization for sustainable information delivery using PD-NOMA deep reinforcement learning." Wireless Networks (2024): 1-21.
[16] Somarin, A. M., Barari, M., & Zarrabi, H. (2018). Big data based self-optimization networking in next generation mobile networks. Wireless Personal Communications, 101(3), 1499-1518.
[17] Kuang, Shuhong, Jiyong Zhang, and Amin Mohajer. "Reliable information delivery and dynamic link utilization in MANET cloud using deep reinforcement learning." Transactions on Emerging Telecommunications Technologies 35, no. 9 (2024): e5028.
[18] Hua, Yuxiu, Rongpeng Li, Zhifeng Zhao, Xianfu Chen, and Honggang Zhang. "GAN-powered deep distributional reinforcement learning for resource management in network slicing." IEEE Journal on Selected Areas in Communications 38, no. 2 (2019): 334-349.
[19] X. Qin, Z. Song, Y. Hao, and X. Sun, "Joint Resource allocation and trajectory optimization for multi-uav-assisted multi-access mobile edge computing," IEEE Wireless Communications Letters, vol. 10, no. 7, pp. 1400–1404, 2021.
[20] Wang, Qianxing, Wei Li, and Amin Mohajer. "Load-aware continuous-time optimization for multi-agent systems: Toward dynamic resource allocation and real-time adaptability." Computer Networks 250 (2024): 110526.
[21] H. Hu, Z. Chen, F. Zhou, Z. Han, and H. Zhu, "Joint Resource and trajectory optimization for heterogeneous-uavs enabled aerial-ground cooperative computing networks," IEEE Transactions on Vehicular Technology, vol. 72, no. 6, pp. 7119–7133, 2023.
[22] Mirzaei, A., Barari, M., & Zarrabi, H. (2019). Efficient resource management for non-orthogonal multiple access: A novel approach towards green hetnets. Intelligent Data Analysis, 23(2), 425-447.
[23] Gu, LiFen, and Amin Mohajer. "Joint throughput maximization, interference cancellation, and power efficiency for multi-IRS-empowered UAV communications." Signal, Image and Video Processing 18, no. 5 (2024): 4029-4043.
[24] G. Chen, Q. Wu, R. Liu, J. Wu, and C. Fang, "Irs aided mec systems with binary offloading: A unified framework for dynamic irs beamforming,"IEEE Journal on Selected Areas in Communications, vol. 41, no. 2, pp. 349–365, 2023.
[25] X. Li, Y. Qin, J. Huo, and W. Huangfu, "Computation offloading and trajectory planning of multi-uav-enabled mec: A knowledge-assisted multiagent reinforcement learning approach, IEEE Internet of Things Journal, 2023.
[26] Yang, Ting, Jiabao Sun, and Amin Mohajer. "Queue stability and dynamic throughput maximization in multi-agent heterogeneous wireless networks." Wireless Networks (2024): 1-27.
[27] Mirzaei, A., & Rahimi, A. (2019). A Novel Approach for Cluster Self-Optimization Using Big Data Analytics. Information Systems & Telecommunication, 50.
[28] Y. Gu, C. Yin, Y. Guo, B. Xia, and Z. Chen, "Communicationcomputation- aware user association in mec hetnets: A meta-analysis," IEEE Transactions on Wireless Communications, vol. 22, no. 9, pp. 6090–6105, 2023.
[29] Zhang, Qi, Zhigang Li, Zhenteng Qin, Xiaochuan Sun, and Haijun Zhang. "Temporal Feature-Enhanced Deep Reinforcement Learning for RAN Slicing with User Mobility." IEEE Communications Letters (2023).
[30] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in uav-enabled wireless-powered mobile-edge computing systems," IEEE Journal on Selected Areas in Communications, vol. 36, no. 9, pp. 1927–1941, 2018.
[31] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao, and G. Y. Li, "Joint offloading and trajectory design for uav-enabled mobile edge computing systems,"IEEE Internet of Things Journal, vol. 6, no. 2, pp. 1879–1892, 2019.
[32] Zhao, Zhongyong, Yu Chen, Jiangnan Liu, Yingying Cheng, Chao Tang, and Chenguo Yao. "Evaluation of operating state for smart electricity meters based on transformer–encoder–BiLSTM." IEEE Transactions on Industrial Informatics 19, no. 3 (2022): 2409-2420.
[33] Mohajer, Amin, Maryam Bavaghar, Rashin Saboor, and Ali Payandeh. "Secure dominating set-based routing protocol in MANET: Using reputation." In 2013 10th International ISC Conference on Information Security and Cryptology (ISCISC), pp. 1-7. IEEE, 2013.
[34] Y. Xu, T. Zhang, Y. Liu, D. Yang, L. Xiao, and M. Tao, "Cellular connected multi-uav mec networks: An online stochastic optimization approach," IEEE Transactions on Communications, vol. 70, no. 10, pp. 6630–6647, 2022.
[35] Nemati, Z., Mohammadi, A., Bayat, A., & Mirzaei, A. (2024). Metaheuristic and Data Mining Algorithms-based Feature Selection Approach for Anomaly Detection. IETE Journal of Research, 1-15.
[36] Li, Rongpeng, Chujie Wang, Zhifeng Zhao, Rongbin Guo, and Honggang Zhang. "The LSTM-based advantage actor-critic learning for resource management in network slicing with user mobility." IEEE Communications Letters 24, no. 9 (2020): 2005-2009.
[37] L. Zhang, J. Li, Y. Wang, Z. Chen, Q. Liu, and Y. Sun, "Task offloading and trajectory control for uav-assisted mobile edge computing using deep reinforcement learning," IEEE Access, vol. 9, pp. 53 708–53 719, 2021.
[38] X. Zhang, J. Zhang, J. Xiong, L. Zhou, J. Wei, and H. Li, "Energy efficient multi-uav-enabled multiaccess edge computing incorporating noma," IEEE Internet of Things Journal, vol. 7, no. 6, pp. 5613–5627, 2020.

[39] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and L. Hanzo, "Multiagent deep reinforcement learning-based trajectory planning for multiuav assisted mobile edge computing," IEEE Transactions on Cognitive Communications and Networking, vol. 7, no. 1, pp. 73–84, 2021.

[40] T. Zhang, Y. Xu, J. Loo, D. Yang, L. Xiao, and Y. Zhao, "Joint computation and communication design for uav-assisted mobile edge computing in iot," IEEE Transactions on Industrial Informatics, vol. 16, no. 8, pp. 5505–5516, 2020.

[41] Z. Liu, X. Tan, M. Wen, S. Wang, C. Liang, and Q. Zhao, "An energy efficient selection mechanism of relay and edge computing in uavassisted cellular networks," IEEE Transactions on Green Communications and Networking, vol. 5, no. 3, pp. 1306–1318, 2021.

[42] Yan, Dandan, Benjamin K. Ng, Wei Ke, and Chan-Tong Lam. "Deep reinforcement learning based resource allocation for network slicing with massive MIMO." IEEE Access (2023).

[43] C.-Y. Hsieh, Y. Ren, and J.-C. Chen, "Edge-cloud offloading: Knapsack potential game in 5g multi-access edge computing," IEEE Transactions on Wireless Communications, vol. 22, no. 4, pp. 3124–3136, 2023.

[44] N. Zhao, C. Xu, W. Zhang, S. Yang, G.-M. Muntean, and F. Zhou,"5g-enabled uav-to community offloading: Joint trajectory design and task scheduling," IEEE Journal on Selected Areas in Communications, vol. 39, no. 11, pp. 3306–3320, 2021.

[45] H. Guo and J. Liu, "Uav-enhanced intelligent offloading for internet of things at the edge, IEEE Transactions on Industrial Informatics, vol. 16, no. 4, pp. 2737–2746, 2020.

[46] Wang, Zhaoying, Yifei Wei, F. Richard Yu, and Zhu Han. "Utility optimization for resource allocation in multi-access edge network slicing: A twin-actor deep deterministic policy gradient approach." IEEE Transactions on Wireless Communications 21, no. 8 (2022): 5842-5856.

[47] X. Qin, Z. Song, Y. Hao, and X. Sun, "Joint Resource allocation and trajectory optimization for multi-uav-assisted multi-access mobile edge computing," IEEE Wireless Communications Letters, vol. 10, no. 7, pp. 1400–1404, 2021.

[48] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, and X. Shen, "Energyefficient uav-assisted mobile edge computing: Resource allocation and trajectory optimization," IEEE Transactions on Vehicular Technology, vol. 69, no. 3, pp. 3424–3438, 2020.

[49] Wang, Yue, Yu Gu, and Xiaofeng Tao. "Edge network slicing with statistical QoS provisioning." IEEE Wireless Communications Letters 8, no. 5 (2019): 1464-1467.

[50] H. Guo and J. Liu, "Uav-enhanced intelligent offloading for internet of things at the edge, IEEE Transactions on Industrial Informatics, vol. 16, no. 4, pp. 2737–2746, 2020.