This article has been accepted for publication in IEEE Internet of Things Journal. 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/JIOT.2025.3529836

Optimizing UAV-Assisted Vehicular Edge Computing with Age of Information: A SAC-Based Solution

Shidrokh Goudarzi, *Member, IEEE,* Seyed Ahmad Soleymani, *Member, IEEE,* Mohammad Hossein Anisi, *Senior Member, IEEE,* Anish Jindal, *Senior Member, IEEE,* Pei Xiao, *Senior Member, IEEE*

***Abstract*—Edge computing improves the Internet of Vehicles (IoV) by offloading heavy computations from in-vehicle devices to high-capacity edge servers, typically roadside units (RSUs), to ensure rapid response times for intensive and latency-sensitive tasks. However, maintaining quality of service (QoS) remains challenging in dense urban settings and remote areas with limited infrastructure. To address this, we propose an SDN-driven model for UAV-assisted vehicular edge computing (VEC), integrating RSUs and UAVs to provide computing services and gather global network data via an SDN controller. UAVs serve as adaptable platforms for mobile edge computing (MEC), filling gaps left by traditional MEC frameworks in areas with high vehicle density or sparse network resources. An optimal offloading mechanism, designed to minimize the age of information (AoI) while balancing energy consumption and rental costs, is implemented through a soft actor-critic (SAC)-based algorithm that jointly optimizes UAV trajectory, user association, and offloading decisions. Experimental results demonstrate the model's superior performance, achieving up to 87.2% energy savings in energy-limited settings and a 50% reduction in time-sensitive scenarios, consistently outperforming traditional strategies across various task sizes.**

***Index Terms*—Vehicular edge computing, mobile edge computing, soft actor-critic, computation offloading, unmanned aerial vehicle, deep reinforcement learning, and age of information.**

I. Introduction

The increasing adoption of IoV technology is propelling the advancement of the Internet of Things (IoT) and the implementation of modern mobile applications requiring advanced functionalities, such as autonomous navigation and unmanned driving. However, the limited computing power and battery life of vehicles adversely impact the quality of experience for computation-intensive services run on these vehicles, as well as their operational lifespan [1]. The mobile network topology experiences serial communication links for high-speed data transfer, with service requirements and varying QoS conditions that change dynamically.

S. Goudarzi is with the School of Computing and Engineering, University of West London, St Mary's Rd, Ealing, London, W5 5RF, United Kingdom. (e-mail:shidrokh.goudarzi@uwl.ac.uk).

S. A. Soleymani and P. Xiao are with the Institute for Communication Systems (ICS), Home for 5GIC&6GIC, University of Surrey, Guildford, Surrey, GU2 7XH, United Kingdom (e-mail:s.soleymani@surrey.ac.uk; p.xiao@surrey.ac.uk).

M. H. Anisi is with the School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, United Kingdom (e-mail: m.anisi@essex.ac.uk).

A. Jindal is with the Department of Computer Science, Durham University, Durham DH1 3LE, United Kingdom (e-mail: anish.jindal@durham.ac.uk).

Due to limited computing resources in these vehicles, they cannot meet the high reliability and low processing latency requirements of the applications mentioned above. Therefore VEC has been considered as an effective means to solve the problems [2]. By deploying some servers at the edge of the radio access network, vehicle users can offload their computing tasks to nearby RSUs to relieve the computation burdens of corresponding cellular networks and provide users with more efficient and reliable network services [3], [4].

MEC has emerged as a promising approach to leveraging the benefits of heterogeneous IoT applications by bringing diverse cloud resources, such as storage and computational capabilities, closer to the user, particularly to vehicles. MEC represents a paradigm shift, positioning cloud servers proximate to mobile network vehicles, thereby enhancing computing quality and vehicle battery life by offloading computation-intensive processes [5], [6]. However, its application can be challenging in environments with insufficient infrastructure, such as those requiring disaster response, military operations, emergency aid, or in remote areas like forests, mountains, and wetlands. To overcome these limitations, unmanned aerial vehicle (UAV)-enabled MEC has been proposed and developed as an effective alternative [7].

Prior research [8] has predominantly focused on offloading computation and communication tasks from vehicles, without considering the computation time on UAVs. Yet, in practical scenarios, the duration of computation on UAVs is a critical factor that cannot be overlooked. Additionally, while existing studies [9], [10] have explored partial or binary computational tasks, the aspect of continuous task streaming over a fixed period in the context of UAV-enabled MEC has not been thoroughly investigated.

The study in [11] introduces an offloading scheme aimed at reducing the combined cost of energy consumption and delay. This approach integrates joint optimization of task offloading, selection of MEC servers, power distribution, UAV path planning, and CPU frequency allocation. Likewise, [12] seeks to minimize system energy use and latency by optimizing the size of offloaded tasks at each node within a multi-hop edge computing framework.

This work is driven by the utilisation of UAVs as MEC servers in the wireless networking context. With this in mind, the study introduces a system that combines computation offloading and adaptive computing resource allocation, leveraging UAVs as edge servers to deliver edge computing services

Authorized licensed use limited to: University of Durham. Downloaded on January 20,2025 at 12:15:15 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. 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/JIOT.2025.3529836

to vehicles. The primary aim is to minimise overall energy consumption across all vehicles while ensuring efficient task computation for each vehicle within a specified time frame. This is achieved through a holistic approach that jointly tackles queue-based offloading and resource allocation challenges.

The optimisation challenge described is highly complex due to its non-convex and nonlinear characteristics, classifying it as NP-hard. Cooperative evolution, pioneered by Potter and De Jong [13], leverages parallelism through divide-and-conquer, making it well-suited for distributed network settings. This approach exhibits significant scalability, effectively tackling high-dimensional and computationally intensive problems. Recent studies by Chen et al. [14], Omidvar et al. [15], [16], and Mei et al. [17] have introduced novel decomposition methods that adaptively adjust groupings during optimisation. Despite advancements, many methods still adhere to traditional cooperative evolution frameworks, employing uniform strategies like round-robin allocation across subcomponents such as UAV processors for computation tasks.

Recent studies on UAV-assisted VEC systems highlight the importance of computing result freshness for in-vehicle applications requiring real-time processing. Traditional latency metrics fall short of ensuring timely execution, which is critical for accurate decision-making by vehicle users. For example, if vehicle users generate or send tasks infrequently, even if the systems perform well in terms of latency, they will not be able to meet the stringent requirements of real-time computing applications. Therefore, AoI has been proposed [18] and widely studied [19]–[21] as a useful performance metric to measure information freshness. Moreover, all computing services are currently paid, offloading computing tasks would incur to be paid to an operator [22].

In this paper, we focus on reducing AoI, energy consumption, and rental price for computation offloading in UAV-assisted vehicular edge computing networks. Compared with single airborne or ground-based VEC paradigms, our proposed UAV-assisted VEC system offers more flexibility and comprehensive services. In contrast to existing approaches, we propose a solution to address the VEC offloading problem with Deep Reinforcement Learning (DRL) within the complexities of a dynamic environment.

Our approach utilizes software-defined networking (SDN) in a multi-layered data processing system for optimising computational resources. The architecture includes three layers: the cloud center (CC) at the top, UAV-assisted Mobile Edge Computing (U-MEC) servers in the middle, and mobile devices at the bottom. Centralised control over network elements ensures efficient management [5]. SDN partitions and allocates network flows to reduce congestion and latency [23]. It operates with a centralised controller making resource allocation decisions for cloud connectivity [24].

Importantly, different from the single offloading optimisation, a DRL-based approach usually focuses on a long-term offloading performance which is key to the time-varying dynamic systems. Moreover, unlike the traditional Markov decision process (MDP)-based solutions, the DRL-based approaches can learn offloading strategies by directly interacting with the environment without prior knowledge. Meanwhile, deep neural networks (DNNs) in DRL have strong perceptual computation capability and can support large state and action spaces to fulfill an optimisation task.

Since state updates of vehicle tasks are random and accurate information is not available a priori, it is impossible to derive an accurate analytical solution to reduce AoI based on theoretical knowledge such as queuing theory. Moreover, due to the time-accumulative nature of AoI itself, UAV trajectory control and task offloading decisions are tightly coupled, which requires joint optimisation.

Traditional VEC approaches are primarily ground-based, offering limited flexibility in high-mobility environments or areas lacking stable infrastructure, which hinders effective task management and service delivery in dynamic settings like disaster response or remote regions. Optimizing task offloading and resource allocation in real-time across distributed vehicular networks also remains a substantial challenge, as conventional methods often fail to balance computational load and energy efficiency, particularly in high-speed, multi-vehicle interactions. For applications requiring immediate responses and accurate decision-making, traditional latency metrics fall short, as they do not account for "information freshness," or AoI, a critical measure for ensuring timely data availability in autonomous vehicular systems. Despite its relevance, AoI is still underexplored in real-time vehicular edge applications. Addressing these limitations, this study proposes a UAV-assisted VEC system that leverages a DRL approach within the SDN framework. This design focuses on minimizing AoI, energy consumption, and rental costs, offering a scalable, adaptive solution tailored to the dynamic needs of multi-vehicle networks.

Numerous research studies have employed UAVs for their versatile deployment in wireless communication scenarios [25]–[27]. Past studies have investigated the coverage and connectivity in three-dimensional networks potential of UAVs when integrated with cellular networks, where they serve as aerial nodes. Additionally, UAV relaying has been identified as a crucial application capable of enhancing communication coverage effectively [28].

Although various strategies have been developed for deploying UAVs to support vehicles with different data requirements, evolutionary-based approaches for simultaneous resource allocation and deployment are still largely unexplored. We have analysed stochastic task models within the MEC framework, encompassing multiple servers and vehicles, to minimise the long-term weighted average power consumption in the system [29].

While similar issues have been addressed in ground-based cellular networks, the management of mobility in MEC servers has received relatively little attention. Previous studies on UAV deployment in communication systems have primarily focused on their applications as mobile relays [30] or airborne base stations [31]. Moreover, while allocating combined communication and computing resources has been studied extensively in multi-user scenarios with a single server MEC [32], these efforts typically overlooked the computation offloading aspect.

Our work differentiates itself from previous studies [30], [32] in several crucial ways. Firstly, we investigate scenarios

Authorized licensed use limited to: University of Durham. Downloaded on January 20,2025 at 12:15:15 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. 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/JIOT.2025.3529836

where UAV-enabled MEC servers can simultaneously serve multiple vehicles, unlike prior studies that generally assumed the MEC server serves vehicles sequentially. Secondly, our research highlights the cooperative optimisation of task execution between UAVs and vehicles, marking a shift from earlier studies that concentrated exclusively on one approach. Thirdly, we introduce an innovative element by incorporating prioritisation among different UAV processors when allocating computing resources.

While UAVs have been widely studied for enhancing coverage and connectivity, their potential as dynamic edge servers in VEC remains underexplored, as most research emphasizes UAVs as relays or static nodes rather than active servers that support simultaneous, multi-vehicle computational needs. Current VEC frameworks often rely on sequential processing or assume uniform task distribution strategies, which are insufficient for the high-speed, complex requirements of UAV-assisted VEC. This limitation highlights a need for adaptive optimization approaches that prioritize task distribution based on real-time UAV and vehicle demands. Existing offloading strategies also typically employ static or round-robin methods, lacking adaptability to fluctuating network conditions. Although some studies address partial task offloading, there is limited integration of deep reinforcement learning (DRL) for UAV-enabled VEC, despite DRL's unique ability to optimize offloading in dynamic environments without prior knowledge. Unlike prior studies, this work investigates UAV-enabled MEC servers designed to support multiple vehicles concurrently and emphasizes cooperative optimization, marking a key advancement over traditional single-server, static allocation models. Against the above background, the primary contributions of this study are summarised as follows:

1) We propose an SDN-based optimisation framework for UAV-assisted VEC networks, integrating aerial and ground edge computing services. Unlike conventional setups, our approach allows partial computation offloading. The SDN controller facilitates information exchange by collecting global data and distributing it to vehicles. Task buffers at edge nodes and vehicles accommodate the randomness of task arrivals, with a first-come, first-served protocol enabling computation across multiple time slots, enhancing practicality.
2) We minimize AoI to ensure fresh computing results while also minimizing energy consumption and rental costs. The term 'Maximum AoI' (MAoI) is defined as the maximum age of information at any node in the network, which captures information freshness effectively and facilitates Deep Reinforcement Learning (DRL) optimization.
3) Given the dynamic nature of UAV-assisted VEC networks and their complex interdependencies, we address the challenge by transforming it into a reinforcement learning problem based on SAC, which is designed using MDP. Our proposed algorithm efficiently manages trajectory control and offloading allocation in dynamic scenarios with continuous action spaces. It utilises an advanced reinforcement learning framework to enhance training stability and prevent Q-value overestimation.

The paper is structured as follows: Section II reviews relevant literature, contextualising the research within the broader field. In Section III, the system model of the SDN-based UAV-assisted VEC network is introduced, laying the foundation for subsequent discussions. Section IV details the formulation of the optimisation problem, providing a framework for addressing the challenges at hand. The proposed algorithm is presented in Section V, offering a novel approach to managing trajectory control and offloading allocation. Section VI presents simulation results, demonstrating the effectiveness of the proposed method. Finally, Section VII concludes the paper, by summarising key findings and suggesting avenues for future research.

II. Network Model

We introduce a UAV-enabled MEC system, illustrated in Figure 1, where each UAV serves as a MEC server for the vehicles within its coverage area. The system architecture is organized into three layers: the cloud computing layer, the edge computing layer, and the device layer. Within this architecture, the control plane and data plane operate across these layers. The control plane, responsible for managing and coordinating resources, utilizes cellular communications, while the data plane, handling data transmission, relies on dedicated short-range communications.

The SDN controller comprises two primary modules. The first module is responsible for task management, which involves storing all mission data related to the vehicles and determining whether tasks should be processed locally or offloaded to the edge for computation. The second module is the edge server module, which monitors the available memory, CPU resources, and server load. The goal is to navigate the UAV to the locations of the offloading from start point to destination while offloading the vehicle's computing tasks to the MEC server for execution.

In the data plane, there are J deployed RSUs and a rotary-wing UAV as edge nodes (ENs) to provide computing services to I vehicles within their transmission range. In addition, each EN is equipped with a computing server and an SDN switch to communicate with the SDN controller. The ENs servers, denoted by a set of RSUs $\mathcal{J} = \{1, ..., j, ..., J\}$, and another set of ENs, where j = J + 1 indicates the edge server as a UAV, $\mathcal{J}^{'} = \{1, ..., j, ..., J, J+1\}$ are uniformly distributed across the area to cater to the ground vehicles in the current investigation. Moreover, $\mathcal{I} \triangleq \{1, ..., i, ..., I\}$ and $\mathcal{N} \triangleq \{1, ..., n, ..., N\}$ represent the sets of vehicles and time slots, respectively. The set $\mathcal{T} = \{1, ..., t, ..., T\}$ is divided into N time slots, each denoting the time required to accomplish a task, where the duration of each slot is τ, thus $T = \tau N$.

The control plane primarily comprises the SDN controller and the DRL agent, linked via a wired backhaul connection. Global information, including vehicle coordinates and task details, is collected by the SDN controller and transmitted to the DRL agent. Subsequently, the DRL agent processes this data to make offload decisions, which are then communicated to each device through the controller. The UAV uses Frequency

Authorized licensed use limited to: University of Durham. Downloaded on January 20,2025 at 12:15:15 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. 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/JIOT.2025.3529836

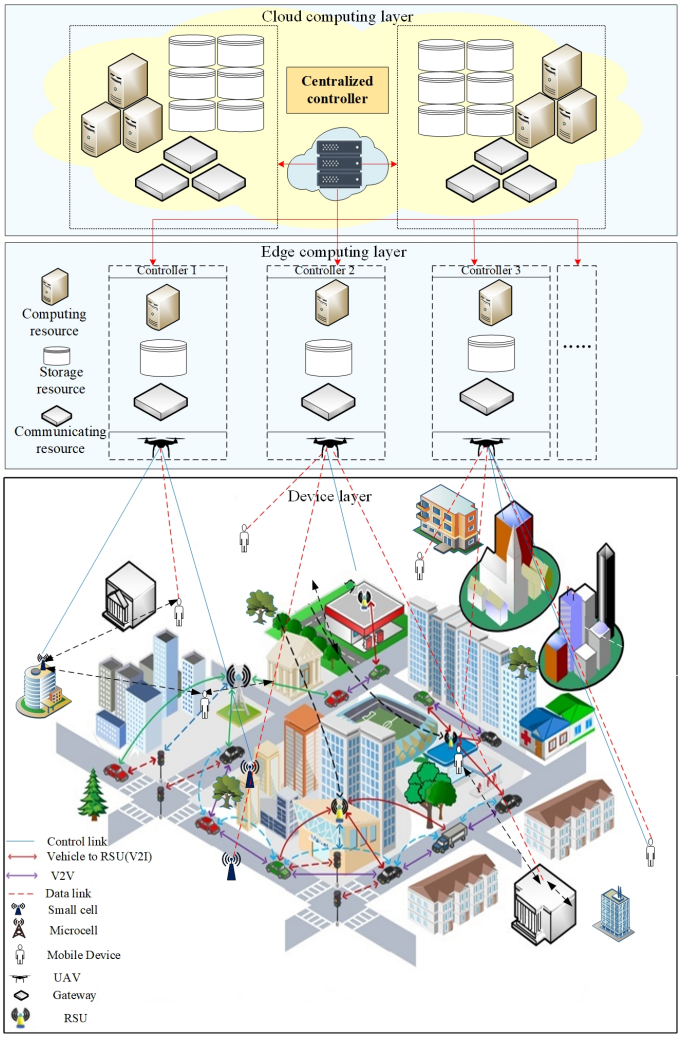

Figure 1: Network model.

Division Multiple Access (FDMA) to allocate bandwidth evenly among vehicles, maintaining a constant altitude $H > 0$. Positions of vehicles and UAVs as edge servers are described in a 3D Cartesian coordinate system. UAV j is located at $\mathbf{u}_j$=(X_j, Y_j, H), and vehicle i is at $\mathbf{m}_i$=$(x_i, y_i, 0)$. The UAV plans resource allocation and position by predicting the 3D coordinates of all vehicles in advance. The flight direction of UAV is determined by the angle of $\theta_u(t) \in [0, 2\pi)$ and the flight velocity of $v_u(t) \in [0, v_{max}]$.

$$X_j(t) = x_u(t-1) + \tau v_u(t) cos(\theta_u(t)), \forall t \in \tau, \quad (1)$$

$$Y_j(t) = y_u(t-1) + \tau v_u(t) sin(\theta_u(t)), \forall t \in \tau, \quad (2)$$

The necessity to consider Line-of-sight (LoS) connections between ground vehicles and the UAV arises due to the UAV's elevated position and low likelihood of dispersion [33]. The influence of LoS routes prevails over NLoS routes [30]. Therefore, the power gain of the channel from vehicle i to UAV j is formulated based on previous research [33].

$$h_{ji} = \frac{g_0}{\|\mathbf{u}_j - \mathbf{m}_i\|^2}, i \in \mathcal{I} \quad (3)$$

where I represents the set of vehicles within the transmission range of the deployed RSUs. Vehicle i divides its computational task at time slot t into two parts: $u_i \geq 0$ bits processed locally and $v_i \geq 0$ bits offloaded to the MEC server via UAV. Here, g_0 represents the channel gain at 1 meter from vehicle i, and $\|\cdot\|$ denotes the l_2 norm.

For the communication links between the RSU and the vehicles, the Rayleigh fading channel model is adopted. The path loss between RSU j and vehicle i in the t-th slot can be expressed (in dB) as

$$PL_{i,j}^{G2G}(t) = 20 \log \left(\frac{\|w_i - w_j(t)\|}{4\pi f} \right) + \eta_{\text{Rayleigh}}, \quad (4)$$

where η is the Rayleigh fading coefficients of G2G channels and f represents the frequency of the communication link.

According to the Shannon theorem, the achieved transmission rate between vehicle i and edge node j in time slot t can be described as

$$R_{i,j}(t) = \beta_{i,j}(t) W_J log_2(1 + (\frac{p_n(t) h_{i,j}(t)}{\sigma_{i,j}^2(t)})) \quad (5)$$

where $W_J = W_R(or W_U)$ is the total uplink bandwidth of each RSU, $p_n(t)$ is the transmit power of vehicle i, $\sigma_{i,j}^2(t) = \beta_{i,j}(t) N_0 W_M$, I denotes the average power of the additive white Gaussian noise (AWGN) at EN j, N_0 is the AWGN spectral density. $h_{i,j}(t) = 10^{(-PL_{i,j}(t)/10)}$, where $PL_{i,j}(t) = PL_{i,j}^{G2G}(t), i \in I$ $(or PL_{i,j}(t) = PL_{i,j}^{A2G}(t), j = J+1)$.

To prevent interference between vehicles and MEC servers during data transmission, we employ the orthogonal frequency division multiple access (OFDMA) technology for communications between the MEC servers and vehicles.

Each UAV allocates a portion of resources, denoted by αj, to macro vehicles, while the remaining portion is allocated to other vehicles of the UAV. The bandwidth demand of each vehicle must not exceed the allocated bandwidth, represented as $(1 - \alpha j)$. It must satisfy $\frac{(1-\alpha j) \cdot Bj}{|\mathcal{I}j|} \leq \max \tilde{B} j$. To ensure QoS, the admission control criterion [34] is employed and is defined as follows:

$$\frac{\max \tilde{B}_j \cdot |\mathcal{I}_j|}{B_j} \leq 1 \quad (6)$$

FDMA enables the UAV to serve $|\mathcal{I}_j|$ vehicles simultaneously. UAVs act as edge servers, providing computing services to vehicles with limited processing power and energy, as illustrated in Figure 1. We assume UAVs have powerful processors for handling complex tasks, supported by previous studies [35].

In mobile topologies, high-speed data transfer and evolving QoS requirements are crucial. Our objective is to optimise the average vehicle rates while maintaining QoS through joint computation offloading and resource allocation. This non-convex mixed-integer problem is tackled using the SAC-Based Optimization technique. The signal-to-noise ratio (SINR) [35], γ_i, for vehicle i is expressed as:

$$\gamma_i = p g_0 / \sigma^2 \quad (7)$$

The data rate achievable by vehicle i from UAV j, denoted as r_i, depends on the transmission power of the UAV (p),

Authorized licensed use limited to: University of Durham. Downloaded on January 20,2025 at 12:15:15 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. 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/JIOT.2025.3529836

noise power (σ^2), and the bandwidth allocated (W) to vehicle i. This is calculated using the Shannon capacity formula as $r_i = Wlog_2(1+\gamma_i)$, where γ_i represents the signal-to-noise ratio.

The connectivity between vehicle i and UAV j is denoted by the variable $a_{i,j}$, where $a_{i,j} = 0$ indicates the task is processed locally on the vehicle itself, and $a_{i,j} = 1$ indicates it is offloaded to UAV j. If r_i represents the available resources for vehicle i, and s_i denotes the required resources, then vehicle i can connect to UAV j if $r_i \geq a_{i,j}S_i$.

UAV deployment aims to maximise vehicle rates and ensure fair coverage with fixed resources $\{S_i|\forall\, i\}$. Tasks are offloaded to the UAV's MEC server for processing, when the execution is finished at the edge server, the result is sent back from the edge server to the vehicle. Vehicles must be within the UAV's coverage area to execute tasks. The distances between vehicles and UAVs are calculated as:

$$d_{i,j} = \sqrt{(x_i - X_j)^2 + (y_i - Y_j)^2 + H^2}, \forall i \in \mathcal{I}, j \in \mathcal{J} \quad (8)$$

The distance between two UAVs is:

$$d_{j_1,j_2} = \sqrt{(X_{j_1} - X_{j_2})^2 + (Y_{j_1} - Y_{j_2})^2}, \quad \forall j_1, j_2 \in \mathcal{J}, j_1 \neq j_2 \quad (9)$$

In this system, $d_{i,j}$ denotes the distance between vehicle i and UAV j. The UAVs are equipped with directional antennas having a fixed beamwidth θ and operate at a constant altitude H with the coverage radius C_j.

Three operational models are under consideration: the UAV hover model, where UAVs maintain a fixed position in the air; the local execution model, where tasks are processed on the vehicle itself; and the MEC execution model, where tasks are offloaded to the MEC server for processing.

A. Hover Dynamics of UAVs

The energy consumption necessary for a UAV to maintain a stationary position over a duration of time, as described by [36], is given by:

$$E^H = P_0 T_h \quad (10)$$

where, T_h represents the duration of hovering, while P_0 denotes the power consumption during hovering.

B. Local computation

The computational capability of vehicle $i \in \mathcal{I}$ is defined by its CPU cycles per second. Each vehicle i manages delay-tolerant tasks shown as $Z_i(t) \triangleq (D_i(t), C_i(t))$, with $D_i(t)$ being the input data size in bits and $C_i(t)$ the required CPU cycles at time slot t [37]. If a task is offloaded to edge node j, vehicle i must be within its coverage.

For local execution, vehicles allocate computing resources, represented by the matrix $\mathbf{L}$, where $l_{i,t}$ indicates the computing capacity of vehicle i in Hz [37]. The local task completion time for vehicle i is then computed as:

$$T_{i,t} = C_i / l_{i,t} \quad (11)$$

The allocation of computing resources to each vehicle by every edge server should not exceed the total resource capacity. The energy consumption for local operations is directly influenced by the required number of CPU cycles [37], formulated as:

$$E_{i,t} = q_0 l_{i,t}^2 D_i C_i \quad (12)$$

where $q_0 > 0$ represents the coefficient of capacitance during the operating processes.

For $U_i(t)$ with the local computation ratio $(1 - o_{i,j}(t))$, the local service latency includes the queuing time and computation time calculated by:

$$\tau_i^{local}(t) = \tau_i^{queue}(t) + \tau_i^{comp}(t) \quad (13)$$

$$\tau_i^{queue}(t) = \frac{X_c Q_i(t)}{F_i}, \quad (14)$$

$$\tau_i^{comp}(t) = \frac{(1 - o_n(t)) X_c D_i(t)}{F_i} \quad (15)$$

According to the widely adopted power consumption model as $k_i(F_i)^{v_i}$ where $k_i \geq 0$ is the effective capacity coefficient dependent on the processor chip architecture, and v_i is typically set to 3. Thus, the computation energy consumption of vehicle n in time slot t can be calculated by

$$E_i^{local}(t) = k_i(F_i)^3 \tau_i^{comp}(t) = k_i(F_i)^2(1 - o_i(t)) X_c D_i(t). \quad (16)$$

C. Computation offloading

In our method, each vehicle manages computational tasks based on independently arriving processed data bits $A_i(t)$ at an average rate r_i. Upon arrival, tasks are queued and can either be processed locally or offloaded to an edge node (EN). Local tasks are denoted as $m_i(t)$, while tasks offloaded to an EN are denoted as $o_i(t)$. The processing density ρ indicates the CPU cycles required per bit of computation. The local processing of tasks $m_i(t)$ is given by:

$$m_i(t) = \frac{C_i(t)\tau}{\rho} \quad (17)$$

where $C_i(t)$ represents the CPU cycles used by vehicle i during time slot t of duration τ. The backlog of tasks $Q_i(t)$ at time slot $t+1$ is updated as:

$$Q_i(t+1) = \max\{Q_i(t) - m_i(t) - o_i(t), 0\} + A_i(t) \quad (18)$$

When tasks are offloaded to an EN, the queue length $L_i(t)$ at the EN processor increases as:

$$L_i(t+1) = \max\{L_i(t) - m_i(t), 0\} + o_i(t) \quad (19)$$

The channel selection $x_{i,j,t}$ guides EN j to choose the optimal, least costly option. After estimating the performance of each option j using $\widetilde{\theta}_{i,j,t}$, the optimal option is chosen as:

$$j = \arg\min_j \left\{\widetilde{\theta}_{i,j,t}\right\} \quad (20)$$

The exploration and exploitation balance is controlled by ω in the estimation of $\widetilde{\theta}_{i,j,t}$:

$$\widetilde{\theta}_{i,j,t} = \bar{\theta}_{i,j,t-1} - \omega\sqrt{\frac{2\ln t}{\widehat{x}_{i,j,t-1}}} \quad (21)$$

Authorized licensed use limited to: University of Durham. Downloaded on January 20,2025 at 12:15:15 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. 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/JIOT.2025.3529836

The exploration count $\widehat{x}_{i,j,t}$ is updated as:

$$\widehat{x}_{i,j,t} = \widehat{x}_{i,j,t-1} + x_{i,j,t} \tag{22}$$

Before offloading computational tasks, vehicles transmit input parameters to the EN server on the UAV or RSU. The EN servers process these tasks based on received data and return results to the vehicles upon completion. Energy usage and task execution time during result return are excluded due to smaller output data size compared to input data [36]. The task completion time on an EN server includes transmission and computation times:

$$T_{i,j} = \frac{D_i}{r_{i,j}} + \frac{C_i}{F_{i,j}}, \forall i \in \mathcal{I}, j \in \mathcal{J} \tag{23}$$

Energy consumption for local task completion on the vehicle is determined by:

$$E_i^{vehicle} = \eta_c (F_i)^{v-1} C_i, \forall i \in \mathcal{I} \tag{24}$$

where η_c is the effective switching capacitance and $v = 3$.

Energy consumption for offloading tasks to UAV j is given by:

$$E_{i,j}^{UAV} = P\frac{D_i}{r_{i,j}} + \eta_c (F_{i,j})^{v-1} C_i, \forall i \in \mathcal{I}, j \in \mathcal{J} \tag{25}$$

where P is the transmission power of each vehicle.

The offloading service latency $\tau_i^{\text{offload}}(t)$ includes transmission time $\tau_{i,j}^{\text{trans}}(t)$, queuing time $\tau_{i,j}^{\text{queue}}(t)$, and computation time $\tau_{i,j}^{\text{comp}}(t)$:

$$\tau_i^{offload}(t) = \tau_{i,j}^{trans}(t) + \tau_{i,j}^{queue}(t) + \tau_{i,j}^{comp}(t) \tag{26}$$

The energy consumed by vehicle i for offloading is calculated as:

$$E_{i,j}^{offload}(t) = p_i(t)\tau_{i,j}^{trans}(t) \tag{27}$$

D. UAV Energy Consumption Model

Similarly, the computation energy consumption of the UAV in time slot t can be expressed as:

$$E_u^{comp}(t) = k_u (F_U)^3 \tau_{i,j}^{comp}(t), j = J + 1. \tag{28}$$

According to [38], the dynamic energy consumption of rotary-wing UAV depends on its flight velocity and is given by Eq 29 in which $u_v(t)$ represents the UAV velocity. P_0 and P_i are the blade profile power and induced power in hovering status, respectively. U_{tip} is the tip speed of the rotor blade, v_0 is the mean rotor-induced velocity in hovering status, d_0 is the fuselage drag ratio, ρ_0 is the air density, s_0 is the rotor solidity, and A_0 is the rotor disc area.

$$\begin{aligned} E_u^{\text{dynamic}}(t) =& r\left(P_0\left(1 + \frac{3\left\|v_u(t)\right\|^2}{U_{\text{tip}}^2}\right)\right) \\ &+ P_i\left(\sqrt{1 + \frac{\left\|u_v(t)\right\|^4}{4u_0^4} - \frac{\left\|v_u(t)\right\|^2}{2v_0^2}}\right)^{\frac{1}{2}} \\ &+ \frac{1}{2} d_0 \rho_0 s_0 A_0 \left\|v_u(t)\right\|^3 \end{aligned} \tag{29}$$

When the UAV is hovering, i.e., when the flight speed is 0, the hover energy consumption can be obtained as $E_u^h = \tau(P_0 + P_i)$. Therefore, the total energy consumption of the UAV in time slot t is given by

$$E_u^h(t) = E_u^{comp}(t) + E_u^{dynamic}(t) \tag{30}$$

The following sections will leverage this network model to address specific problems related to task offloading in UAV-assisted VEC networks, focusing on both queue-based strategies and DRL approaches.

III. Problem Formulation

In this section, we introduce a system reward function intended to evaluate the costs related to processing tasks within the system. Subsequently, we define our optimisation challenge, which aims at simultaneous trajectory management and task offloading in a multi-time slot setting under the UAV-enhanced VEC framework, to decrease the cumulative cost of the system.

Utilizing the network model described in Section III, we can systematically approach the problem formulation, which is critical for understanding the constraints and opportunities within UAV-assisted VEC networks.

A. Cost function

The main factors determining the total cost of the system include AoI, energy consumption, and cost of processing units of CPU cycles per second, which will be detailed in the sequel.

AoI is a destination node-centric metric used to assess the freshness of information within a network [38]. In the context of this UAV-assisted VEC network, we denote $\Delta_n(t)$ as the AoI of vehicle n at time slot t, which is defined as the time elapsed between the current time t and the generation time of the latest results received by vehicle n, i.e,

$$\Delta_n(t) = t - g_n(t), \tag{31}$$

where $g_n(t)$ denotes the task generation time of the latest computing results received by vehicle n before time slot t.

In the proposed computation model, with the utilisation of a partial offloading scheme where tasks are computed simultaneously on both ENs and vehicles, the total execution latency to complete task u can be represented as $T_{i,u} = \max\left\{\tau_{i,u}^{local}, \tau_{i,u}^{offload}\right\}$.

If a task is to be executed on a UAV, the vehicle must be within the UAV's coverage area. To ensure fair coverage of vehicles by the UAV, the following constraint needs to be satisfied under the constraints related to computation offloading and resource allocation needs to be satisfied:

$$a_{i,j} d_{i,j} \leq R_j, \forall i \in \mathcal{I}, j \in \mathcal{J} \tag{32}$$

The coverage radius of each UAV is determined by $R_j = H \cdot tan\theta$, where H represents the constant altitude at which the UAVs fly [39]. Assuming $\Delta_n(0) = 0$, the partial age of information for vehicle i is derived as:

$$A_i^{total}(t) = \max\left\{\tau_i^{local}(t), \tau_i^{offload}(t)\right\} + Y_i(t) \tag{33}$$

In the context of the partial offloading scheme, the overall energy consumption for task execution can be articulated as the

Authorized licensed use limited to: University of Durham. Downloaded on January 20,2025 at 12:15:15 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. 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/JIOT.2025.3529836

aggregate of two distinct components: the energy consumed during local computation and the energy consumed during computation offloading, specifically:

$$E_i^{total}(t) = E_i^{local}(t) + \alpha_{i,j}(t) E_{i,j}^{offload}(t) \tag{34}$$

When considering the cost of processing units of CPU cycles per second from the EN server, it's crucial to acknowledge that RSUs and UAVs may possess different computation capabilities, resulting in varied service prices. We denote the unit price (in pence per CPU cycle) of computing resources rented by RSUs as P_R and by UAVs as P_U. Thus, the total price paid by vehicle i to EN j is calculated as:

$$P_i^{total}(t) = P_j o_i(t) X_c D_i(t) \tag{35}$$

Hence, the cost function of vehicle i can be represented as a linear function comprising the aforementioned three costs, expressed as:

$$C_i^{total}(t) = \gamma_A A_i^{total}(t) + \gamma_E E_i^{total}(t) + \gamma_P P_i^{total}(t) \tag{36}$$

where γ_A, γ_E, and γ_P represent the weight coefficients assigned to the AoI cost, energy consumption, and cost of processing units of CPU cycles per second from the EN server, respectively, with the constraint $\gamma_A + \gamma_E + \gamma_P = 1$. The main objective is to optimise UAV trajectory, user association, and offloading allocation to minimise long-term costs for total I vehicles.

We define two sets $A(t) = \{\alpha_{i,j}(t)\}$ for i in I and J in $J^{'}$, and $O(t) = \{o_i(t)\}$ for i in I. Then, the problem of minimising the total cost can be formulated as:

$$OP_1 : \min_{w_u(t), A(t), O(t)} \sum_{t\in\tau} \sum_{i\in I} C_i^{total}(t) \tag{37}$$

$$s.t. \left\| w_i(t+1) - w_i(t) \right\| = v_i(t)\tau, \forall i \in I, \forall t \in \tau \tag{38}$$

$$\left\| w_u(t+1) - w_u(t) \right\| \leq v_{max}\tau, \forall t \in \tau \tag{39}$$

$$v_u(t) \in [0, v_{max}], \theta_u \in [0, 2\pi), \forall\, t \in \tau \tag{40}$$

$$\alpha_{i.j}(t) \in \{0,1\}, \forall\, j \in J^{'}, \forall\, i \in I, \forall\, t \in \tau \tag{41}$$

$$\sum_{j=1}^{J+1} \alpha_{i,j} = 1, \forall\, i \in I, \forall\, t \in \tau \tag{42}$$

$$\sum_{t\in\tau} E_u(t) \leqslant E_U^{max} \tag{43}$$

$$a_{i,j} d_{i,j} \leq R_j, \forall i \in \mathcal{I}, j \in \mathcal{J} \tag{44}$$

Constraint (38) defines the trajectory of a vehicle traveling at a variable speed $v_u(t)$. Constraints (39) and (40) ensure that the UAV flight speed remains within the prescribed maximum flight speed. Constraints (41)-(42) govern the user association variable, ensuring that each vehicle selects only one EN per time slot for offloading, with limitations on the maximum number of associations per EN. Constraint (41) guarantees that each task can either be processed locally or offloaded to an EN. Constraint (43) restricts the UAV's energy consumption to be no greater than its available energy consumption, with E_U^{max} representing the maximum energy stored by the UAV. Additionally, constraint (44) specifies that if a task is offloaded to UAV j, vehicle i must be situated within the coverage area of UAV j. The coverage radius of each UAV is denoted as R_j, and the UAVs are equipped with directional antennas featuring a fixed beamwidth θ.

It's worth noting that problem (37) is intractable as it involves a non-smooth and non-convex objective function due to the presence of the non-differentiable function. Additionally, the set A(t) contains binary discrete variables, rendering the feasible set of the optimisation problem non-convex. Given that OP_1 is a mixed-integer non-linear programming (MINLP) NP-hard problem, obtaining a closed-form optimal solution using traditional methods is challenging.

IV. Queue Based Offloading for Task Offloading in UAV-Assisted VEC Networks

As outlined in the network model, the efficiency of task offloading mechanisms is contingent upon the network's architecture and dynamics. This section explores how these characteristics affect queue-based offloading in UAV-assisted VEC networks. In a queue-based system, vehicle i's computation tasks depend on $A_i(t)$, the data processed at time slot t. Tasks arrive independently at intervals with an average rate r_i. Vehicles maintain a queue for incoming tasks, managed in a buffer where they're processed in order. Tasks are categorized as local $m_i(t)$ or offloaded $o_i(t)$. The processing density ρ denotes CPU cycles per bit. The local queue size $m_i(t)$ at vehicle i. reflects stored data.

$$m_i(t) = \frac{C_i(t)\tau}{\rho} \tag{45}$$

At time slot t, where each slot has a duration of τ, $C_i(t)$ denotes the number of CPU cycles of vehicle i. The task queue backlog of vehicle i at time slot t, denoted as $Q_i(t)$, is updated as follows:

$$Q_i(t+1) = \max\{Q_i(t) - m_i(t) - o_i(t), 0\} + A_i(t) \tag{46}$$

When tasks are offloaded, the UAV processor must accommodate the tasks in numerous parallel buffers. Given the typically superior computational capacity of the UAV processor, it is well-equipped to handle offloaded tasks. The increase in the queue length $L_i(t)$ at the UAV processor for vehicle i is expressed as follows:

$$L_i(t+1) = \max\{L_i(t) - m_i(t), 0\} + o_i(t) \tag{47}$$

Initially, queue lengths and indicator values are zero. Decisions are made in each slot. The channel selection indicator is $x_{i,j,t}$. UAV processor j selects the option, updating queues and setting $\theta_{i,j,t} = 1$ iteratively until $t > T$. It uses $\bar{\theta}_{i,j,t-1}$ to estimate $\theta_{i,j,t}$ for consistent optimal choices and $\widehat{x}_{i,j,t-1}$ to balance exploration and exploitation. This strategy aids vehicle i in making informed decisions while $\widehat{x}_{i,j,t-1}$ tracks option

Authorized licensed use limited to: University of Durham. Downloaded on January 20,2025 at 12:15:15 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. 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/JIOT.2025.3529836

Algorithm 1 Queue Management Algorithm

1: **Input:** ω, τ_j, $\theta_{i,j,t}$, $x_{i,j,t}$
2: **Output:** Optimal offloading decisions $\widetilde{\theta}_{i,j,t}$
3: // Initialization
4: Initialize: $\bar{\theta}_{i,j,t} = 0$, $\widehat{x}_{i,j,t} = 0$
5: **while** $t \leq T$ **do**
6: // Estimation and Decision Making
7: Compute estimation $\widetilde{\theta}_{i,j,t}$ using (48)
8: Select optimal option j based on (49)
9: Update exploration count $\widehat{x}_{i,j,t}$ using (50)
10: // Task Updates
11: Update local task $m_i(t)$ according to (45)
12: Update task backlog $Q_i(t)$ using (46)
13: Update EN task backlog $L_i(t)$ according to (47)
14: Increment t
15: **end while**

selection frequency and $\widetilde{\theta}$ estimates throughput for each option j in slot t.

$$\widetilde{\theta}_{i,j,t} = \bar{\theta}_{i,j,t-1} - \omega \sqrt{\frac{2 \ln t}{\widehat{x}_{i,j,t-1}}} \tag{48}$$

The first term in the above equation indicates the effectiveness of option j, with ω being the exploration-exploitation trade-off weight. A higher ω value signifies a greater emphasis on exploration. The minimum estimated value, derived after evaluating $\widetilde{\theta}_{i,j,t}$ across all subchannels of UAV j, is defined as follows:

$$j = \arg\min_{j} \left\{ \widetilde{\theta}_{i,j,t} \right\} \tag{49}$$

Accordingly, $\widehat{x}_{i,j,t}$ is updated as:

$$\widehat{x}_{i,j,t} = \widehat{x}_{i,j,t-1} + x_{i,j,t} \tag{50}$$

Algorithm 1 outlines the process of queue management, task offloading decision-making, and queue updates. The UAV processor j selects the optimal task offloading strategy based on current information, encompassing both local and non-local data, aiming for maximum success probability. This process iterates from lines 6 to 12 until $t > T$. The algorithm efficiently handles computation tasks by dynamically adjusting queue operations and adapting to changes in data backlog and service conditions. The UAV processor enhances task execution latency by utilising parallel buffers and efficiently managing offloaded tasks to improve overall data processing rates.

V. DRL for Task Offloading in UAV-Assisted VEC Networks

This section proposes an RL-based joint trajectory control and offloading allocation algorithm to solve the optimisation problem. Building on the principles of the network model, we explore how DRL techniques can optimize task offloading strategies by considering the network's state dynamics as defined in Section III.

A. MDP Formulation

MDP serves as a standardised framework for resolving sequential decision problems [40]. Given the dynamic evolution of buffer queues in vehicles and ENs, the continuous UAV flight trajectories, varying states of channel gains, vehicle locations, and computing tasks at each time slot in the considered networks satisfy the Markov property. Consequently, we transform the optimisation problem into a model-free MDP, considering the unknown changing environment where obtaining state transition probabilities is challenging. A model-free MDP typically comprises a 3-tuple (S, A, R), where S denotes the system state space, A denotes the action space, and R denotes the reward function.

In our system, during time slot t, the DRL agent acquires global information via the SDN controller, defining the current environment state as $(s_t) \in S$. The agent selects an action from A according to a policy $\pi(a_t|s_t)$, interacts with the environment, and receives an immediate reward $r_t(s_t, a_t) \in R$. Subsequently, the environment state transitions to the next state $s_t + 1 \in S$. Continuously improving its policy based on obtained rewards, the agent learns iteratively through constant interaction with the environment until an optimal policy is achieved. For the optimisation problem, the system's state space, action space, and immediate reward function are defined in the sequel.

1) State space S: At the onset of each time slot, the SDN controller transmits the periodically collected global dynamic information to the agent from the UAV-assisted VEC network. This information is regarded as the state of the environment observed by the agent. We define the state at time slot t as:

$$s_t = \{W(t), U(t), Y(t), Q(t), H(t), E_U^{res}(t)\}, \tag{51}$$

where $W(t), U(t), Y(t), Q(t), H(t), E_U^{res}(t)$ are defined as: $W(t) = [W_1(t), W_2(t), ..., W_I(t)]$ represents the set of real-time coordinates of all the vehicles and UAVs in time slot t. Given that the altitude of the UAV remains constant during its flight, we only need to consider its horizontal coordinates. Therefore, $W(t)$ is a vector of size $2(I + 1)$.

$U(t) = [U_1(t), U_2(t), ..., U_I(t)]$ represents the information of all the tasks generated in the time slot t, where $U(t)$ is a vector of size $2I$. If vehicle i does not generate a task in time slot t, $U_i(t) = 0$. $Y(t) = [Y_1(t), Y_2(t), ..., Y_I(t)]$ is an N-dimensional vector containing the time interval for all the vehicles to generate tasks. $Q(t) = [B_1(t), \cdots, B_N(t), Q_1(t), \cdots, Q_J(t), Q_{J+1}(t)]$ is an (N + M + 1)-dimensional vector containing the backlog of computation queues for all vehicles and ENs. $H(t) = [h_{1,1}(t), \cdots, h_{i,j}(t), \cdots, h_{I,J+1}(t)]$ is a $I(J+1)$-dimensional vector containing the channel gain between the vehicles and ENs. $E_u^{res}(t)$ represents the residual energy of the UAV in time slot t, which is updated according to the formula $E_u^{res}(t+1) = max\{E_u^{res}(t) - E_u(t), 0\}$, with $E_u^{res}(0) = E_U^{max}$.

2) Action space A: In time slot t, the DRN agent takes a collective action, determined by the observed system state s_t, which includes the following three actions:

Adjustment of the UAV trajectory, denoted as $w_u(t)$: The UAV manages its flight path by altering both its velocity and

Authorized licensed use limited to: University of Durham. Downloaded on January 20,2025 at 12:15:15 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. 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/JIOT.2025.3529836

angle, represented as $\{v_u(t), \theta_u(t)\}$. User association, noted as $A(t)$: The utilised DRL algorithm operates within a continuous action space. Given that $\alpha_{i,j}(t) \in \{0, 1\}$ is discrete, we convert it into a continuous variable $\alpha_i(t) \in [0, 1]$. This variable, named the offloading allocation variable, ensures consistency. The decision for task offloading, denoted as $o_i(t)$, involves selecting the offloading ratio on(t) from the continuous range $[0, 1]$. To simplify the action space, we introduce the following definition:

$$o_i(t) = \begin{Bmatrix} 3\alpha(t) - \lfloor 3\alpha(t) \rfloor & , \text{if } \alpha(t) \in \left(\frac{1}{3}, \frac{2}{3}\right) \bigcup \left(\frac{2}{3}, 1\right) \\ 1, & \text{if } \alpha_i(t) \in \left(\frac{2}{3}, 1\right) \\ 0, & \text{otherwise} \end{Bmatrix} \tag{52}$$

Thus, the action is determined solely by the trajectory control variables and the offloading allocation variables, represented as:

$$o_t = \{v_u(t), \theta_u(t), \alpha_1(t), ..., \alpha_i(t), ..., \alpha_N(t)\} \tag{53}$$

3) Reward function R: To minimise the overall system cost, we establish an immediate reward function inversely correlated with the cost function. Moreover, to adhere to the constraints (41) and (42) outlined in the optimisation problem, we introduce two penalty terms. Consequently, the immediate reward function is defined as:

$$r_t = -\sum_{i \in I} C_i^{total}(t) - \phi_1(t) - \sum_{i \in I} \phi_{2i}(t) - \phi_3(t) \tag{54}$$

where

$$\phi_1(t) = \begin{Bmatrix} c_1, & \text{if } (41) \text{ or } (42) \text{ not satisfy} \\ 0, & \text{otherwise} \end{Bmatrix} \tag{55}$$

$$\phi_2(t) = \begin{Bmatrix} c_2, & \max\left\{\tau_i^{local}(t), \tau_i^{offload}(t)\right\} \geq T_i^{max}(t) \\ 0, & \text{otherwise} \end{Bmatrix} \tag{56}$$

$$\phi_3(t) = \begin{Bmatrix} c_3, & E_u^{res}(t) < 0 \\ 0, & \text{otherwise} \end{Bmatrix} \tag{57}$$

where c_1, c_2, and c_3 are positive constants representing penalty terms.

The MDP aims to determine the optimal policy $\pi : S \rightarrow A$, to maximise the expected long-term discounted cumulative reward.

$$\max_{\pi} \lim_{k \rightarrow \infty} E_\pi \left[\sum_{t=0}^{k} C^t r_t\right] \tag{58}$$

where $E_\pi(.)$ represents the expected value obtained by following the policy π. The parameter C, which ranges from 0 to 1, serves as a discount factor, reflecting the influence of future rewards on the current state. A value close to 0 indicates the agent prioritises immediate gains without considering future benefits, whereas a value near 1 signifies equal importance placed on long-term rewards. Consequently, utilising the MDP as mentioned in the earlier model, we can reformulate the optimisation problem as:

$$\max_{v_u(t), a(t)_{i \in I}} \frac{1}{T} \sum_{t=1}^{T} C^t r_t \tag{59}$$

The optimisation problem represents an unconstrained model-free MDP problem that can be addressed using our proposed DRL-based algorithm, as detailed in the subsequent discussion.

B. ACTOR–CRITIC-BASED TASK OFFLOADING

We employ the Soft Actor-Critic (SAC) algorithm in the proposed DRL-based algorithm to train our agent for optimal UAV trajectory control and offloading allocation. SAC is a deep reinforcement learning algorithm designed for continuous action spaces. This selection is based on SAC's ability to efficiently handle sequential decision problems in environments with high-dimensional state spaces and continuous action spaces, ensuring swift, stable, and accurate solutions.

The SAC algorithm is constructed under the framework of MDP, where an agent interacts with an environment characterised by states, actions, and rewards. This interaction occurs over discrete time steps. At each step, the agent observes the current state, takes an action, receives a reward, and transitions to the next state. Through this process, the agent continually refines its policy to maximise long-term rewards. SAC specifically focuses on regulating the entropy of the policy, which encourages exploration. It employs maximum entropy reinforcement learning to simultaneously maximise both the expected reward and entropy, aiming to find the optimal policy π. This policy is probabilistic in nature, mapping states to action probability distributions. SAC relies on the Bellman equation, a core principle in MDPs, to establish relationships between state-action values and state values. In the context of air-to-ground networks, these relationships dictate how actions and states influence future rewards and decisions.

In accordance with the Bellman equation [21], the relationship between the state s, action a, and time t for the given air-to-ground network is expressed as:

$$Q^\pi(s_t, a_t) = R_{(s_t, a_t)} + \gamma \mathbb{E}_{s_{t+1} \sim p}\left[V\left(s_{t+1}\right)\right] \tag{60}$$

$$V_{s_t} = \underset{a_t \sim \pi}{E}\left[Q(s_t, a_t) - \log \pi\left(a_t | s_t\right)\right] \tag{61}$$

Here, p represents the trajectory distribution generated by the policy π, and $R_{(s_t, a_t)}$ is the reward obtained from the state-action pair at time t. We define $Q^\pi(s_t, a_t)$ as $Q_\theta(s_t, a_t)$ in the deep neural network (DNN), where θ denotes the network parameters.

The network architecture includes actor networks (μ) and two sets of critic networks ($Q1, Q2$), with each set comprising both online and target networks. These networks have distinct parameter sets, denoted as $(\theta_\mu, \theta'_\mu, \theta_{c1}, \theta_{c2}, \theta'_{c1}, \theta'_{c2})$. The parameters of the Q-function are subject to change, and the actor and critic components undergo updates based on the information stored in the replay buffer, which includes actions and immediate rewards. Accordingly, the parameters are optimised by minimising a specific loss function, according to [40].

$$Q_{val} = \left(Q_\theta(s_t, a_t) - Q'_{\theta'}(s_t, a_t)\right)^2 \tag{62}$$

$$J_Q(\theta) = \underset{(s_t, a_t) \sim B}{E}\left[\frac{1}{2} Q_{val}\right] \tag{63}$$

Authorized licensed use limited to: University of Durham. Downloaded on January 20,2025 at 12:15:15 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. 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/JIOT.2025.3529836

where $Q_\theta(s_t, a_t)$ denotes the soft Q-value, B denotes the replay buffer, and

$$Q^{'}_{\theta'}(s_t, a_t) = R_{(s_t,a_t)} + \gamma \underset{s_{t+1}\sim p}{E}[V_{\theta'}(s_{t+1})] \tag{64}$$

The target action-value function, denoted by $\theta^{'}$, is introduced to stabilise the iterations for the action-value function. It is updated using an exponentially weighted moving average of the parameters θ. This process helps smooth out the updates and stabilise the learning process. At the onset of each episode, the algorithm initialises the positions of vehicles and UAVs, along with queue backlogs, UAV energy levels, and channel states. Additionally, it initialises six sub-networks and a replay buffer B. During each time step t, vehicles, RSUs, and the UAV share their states with the agent based on global information provided by the SDN controller. Subsequently, the online actor network generates control actions, represented as $a_t = \mu(s_t; \theta\mu) + \epsilon$, where $\epsilon \sim N(0, \sigma)$ denotes Gaussian noise facilitating exploration. Based on these actions, the UAV adjusts its position to $w_u(t)$, vehicles offload tasks to ENs, and the system transitions from state s_t to s_{t+1}, receiving an immediate reward r_t computed according to equation (33).

To address the computational complexity of the Soft Actor-Critic (SAC) algorithm in the UAV-assisted VEC system, the primary cost factors are updating the actor and critic networks, sampling and processing mini-batches, and calculating target Q-values. The complexity of updating the actor and two critic networks scales with the network size $O(p)$, where p is the number of parameters. Mini-batch sampling and Q-value calculations add $O(|N_b|)$, where $|N_b|$ is the mini-batch size. Given that these operations are repeated over N_{steps} steps per episode and N_{episodes} total episodes, the overall computational complexity is approximately:

$$O(N_{\text{episodes}} \cdot N_{\text{steps}} \cdot (p + |N_b|)) \tag{65}$$

This reflects the scaling of computational costs with the number of training episodes, steps per episode, network size, and mini-batch size.

Algorithm 2 Soft Actor-Critic (SAC) Algorithm for UAV-assisted VEC System

Input: UAV initial state s_0, user task demands, environmental parameters, discount factor ζ, learning rates $\xi_\mu, \xi_{c1}, \xi_{c2}$, exploration noise standard deviation σ, mini-batch size $|N_b|$, maximum episodes N_{episodes}, maximum steps per episode N_{steps}, target network update rate ψ
Output: Optimized actor network parameters θ_μ for UAV trajectory and task offloading policies, updated critic network parameters θ_{c1}, θ_{c2} for state-action value estimation
Initialize actor network μ and two sets of critic networks $Q1$ and $Q2$ with random parameters $\theta_\mu, \theta_{c1}, \theta_{c2}$
Initialize target networks $\theta'_\mu \leftarrow \theta_\mu$, $\theta'_{c1} \leftarrow \theta_{c1}$, $\theta'_{c2} \leftarrow \theta_{c2}$
Initialize replay buffer B, mini-batch size $|N_b|$, discount factor ζ, learning rates $\xi_\mu, \xi_{c1}, \xi_{c2}$, updated rate ψ, maximum episodes N_{episodes}, and maximum steps N_{steps}
for episode $= 1$ **to** N_{episodes} **do**
  Randomly initialize system environment and observe the initial state s_0
  for $t = 1$ **to** N_{steps} **do**
    Select action with exploration noise:
    $a_t = \mu(s_t; \theta_\mu) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$
    Obtain immediate reward r_t and next state s_{t+1}
    Store experience tuple (s_t, a_t, r_t, s_{t+1}) in B
    if B is full **then**
      Randomly sample a mini-batch of N_b tuples (s_i, a_i, r_i, s_{i+1}) from B
      Compute target action: $a'_{t+1} = \mu'(s_{t+1}; \theta'_\mu) + \epsilon'$, $\epsilon' \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c)$
      Update target Q-values: $y_i = r_i + \zeta \min(Q'_1(s_{i+1}, a'_{t+1}; \theta'_{c1}), Q'_2(s_{i+1}, a'_{t+1}; \theta'_{c2}))$
      Calculate the TD-error and update critic networks θ_{c1} and θ_{c2}
      if $\text{mod}(t, 2) = 0$ **then**
        Update actor network by minimizing the SAC loss
        Soft update target networks
      end if
    end if
  end for
end for

VI. Numerical Evaluation

In this section, we showcase numerical simulation results to evaluate the effectiveness of the proposed Soft Actor-Critic (SAC) algorithm in a UAV-assisted VEC system. Our simulations were conducted on a computer with an Intel i7-12700k CPU and an NVIDIA RTX 3080 GPU with 12GB of video memory. All neural networks were trained using Python 3.9.12 and Tensorflow-gpu 2.9.1.

Algorithm 2 comprises six identical neural networks, each consisting of two fully connected hidden layers with $[600, 400]$ neurons. The hidden layers of the networks utilise the rectified linear unit (RLU) as the activation function. In contrast, the output layers of the actor networks employ Tanh to constrain the range of output action values to $[1, -1]$. We conducted experiments in different simulated road scenarios to ensure the diversity of traffic feature data. In the first scenario, a 600m two-way lane with 3 RSUs is uniformly deployed, each with a coverage radius of 100m. The second scenario includes an area comprising one 600m east-west lane and two 400m north-south lanes, with three RSUs deployed at (100, 100), (300, 300), and (500, 100). All vehicles are randomly distributed on this roadway and travel at speeds ranging from 10 to 20 m/s. The duration of each time slot is set to 0.05s. At the beginning of each time slot, vehicles randomly generate computation-intensive tasks. The task arrival rates are defined by a Poisson distribution with mean arrival rates set to 0.2, 0.4, 0.6, 0.8, and 1.0 tasks per second. These values represent different traffic conditions, where lower rates correspond to lower task generation frequencies, and higher rates simulate more computationally demanding scenarios with increased task arrival frequency.

We evaluate the performance of the SAC proposed algorithm under various initial conditions and dimensions, com-

Authorized licensed use limited to: University of Durham. Downloaded on January 20,2025 at 12:15:15 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. 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/JIOT.2025.3529836

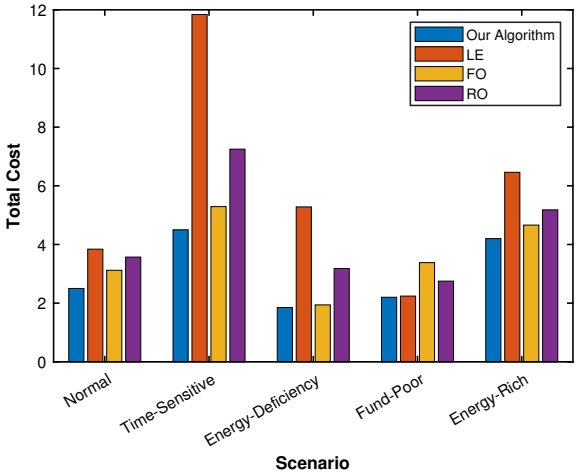

Figure 2: Comparison of the average total cost across various scenarios, illustrating the impact of different configurations on cost performance.

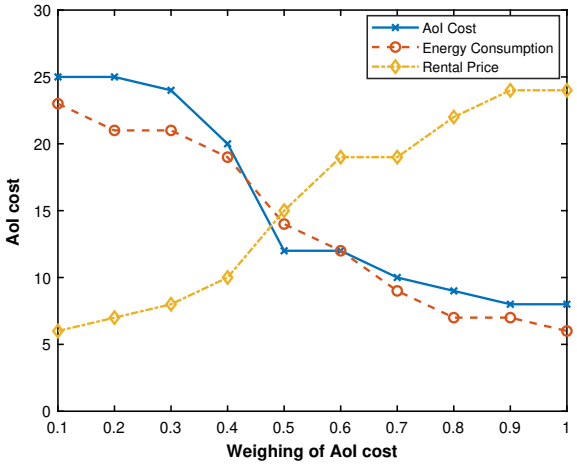

Figure 3: Tradeoff among AoI cost, energy consumption, and rental price across different weightings on AoI, showing how variations impact overall cost performance.

paring it with the following four baseline strategies to verify its effectiveness in achieving optimal trajectory control and offloading allocation:

- Proposed Algorithm for offloading allocation: A scheme based on joint optimisation.
- Local execution (LE): All computing tasks are executed locally.
- Full offloading (FO) [29]: All computing tasks are fully offloaded to the RSUs for execution. An alternative scheme based on the queuing algorithm is used instead of joint optimisation.
- Random offloading (RO) [8]: Computing tasks are randomly assigned to be executed locally, offloaded to the RSUs, or the UAV.

Mathematically, the convergence for FO can be characterized by the time required to reach a steady state where all tasks are processed by the RSUs. The cost reduction in FO follows a linear or sub-linear trajectory because of its lack of dynamic adaptation:

$$C_{FO} = C_{initial} - \lambda.t \tag{66}$$

where λ is the rate of cost reduction per time step, and $C_{initial}$ is the initial system cost. The LE and RO methods do not optimize offloading decisions, and therefore their convergence speeds are significantly slower compared to SAC. The LE method, in particular, does not offload any tasks, resulting in higher system costs and a slow reduction in cost. The RO strategy assigns tasks randomly, leading to inefficient resource utilization and even slower convergence.

The effectiveness of the proposed algorithm is demonstrated by evaluating the system average cost across various application scenarios in, Fig. 2. In the normal scenario, the three sub-costs are equally weighted, i.e., $\lambda_A = 0.33$, $\lambda_E = 0.33$, $\lambda_P = 0.33$. Compared to the LE, FO, and RO strategies, the proposed algorithm reduces the total system cost by 56.6%, 12.4%, and 44.0%, respectively. In the time-sensitive scenario, where real-time information requires frequent updates, we assign a higher weight to the AoI cost, setting $\lambda_A = 0.8$, $\lambda_E = 0.1$, $\lambda_P = 0.1$. The proposed algorithm reduces the system cost by 50% compared to the LE strategy. This significant reduction highlights the superior computation capability of edge servers. Moreover, the cost differences between our algorithm and other strategies are also notable. In the energy-deficiency scenario where system energy consumption is critical, we set $\lambda_A = 0.1$, $\lambda_E = 0.8$, $\lambda_P = 0.1$. Here, the proposed algorithm achieves an 87.2% reduction in cost compared to the LE strategy, owing to the higher energy consumption of local computation. Similarly, in the fund-poor scenario, with priorities on minimising costs, we set $\lambda_A = 0.1$, $\lambda_E = 0.1$, $\lambda_P = 0.8$. Both the proposed algorithm and LE strategy incur lower costs compared to other strategies, with the proposed algorithm emphasising cost minimisation and LE benefiting from rental-free local computing. Meanwhile, the cost of the RO strategy remains intermediate. In the energy-rich scenario, where the onboard energy supply is sufficient, reducing AoI cost and the rental price are prioritised. Hence, we set $\lambda_A = 0.45$, $\lambda_E = 0.1$, $\lambda_P = 0.45$. While the differences in costs between strategies are not as pronounced as in other scenarios due to tradeoffs between AoI cost and rental price, the proposed algorithm still achieves cost reductions of 33.0%, 7.8%, and 17.2% compared to LE, FO, and RO strategies, respectively. Overall, the proposed algorithm demonstrates superior performance across different scenarios, outperforming traditional computation offloading strategies.

Fig. 3 depicts the tradeoff among the AoI cost, energy consumption, rental price, and system cost with varying weights on AoI. As established in the previous section, we maintain the constraint $\lambda_A + \lambda_E + \lambda_P = 1$. Specifically, we set λ_E to 0.1 and vary λ_A from 0.1 to 0.9 with an increment of 0.2, resulting in a corresponding decrease in λ_P.

As λ_A increases, the AoI cost decreases while the rental price increases. This trend is attributed to smaller values of λ_A representing the fund-poor scenario, where the agent

Authorized licensed use limited to: University of Durham. Downloaded on January 20,2025 at 12:15:15 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. 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/JIOT.2025.3529836

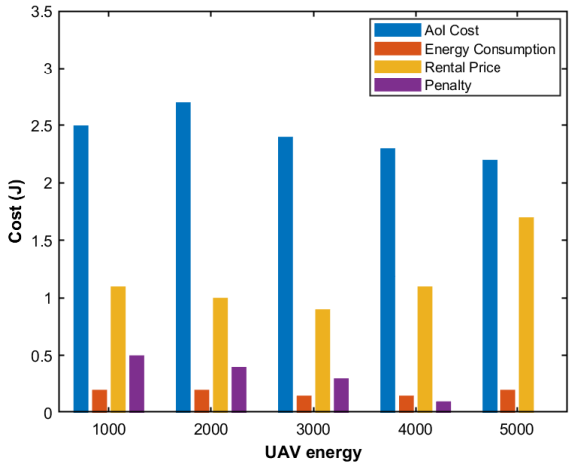

Figure 4: Impact of the available energy of the UAV on the system cost, illustrating how varying energy levels influence the overall cost performance.

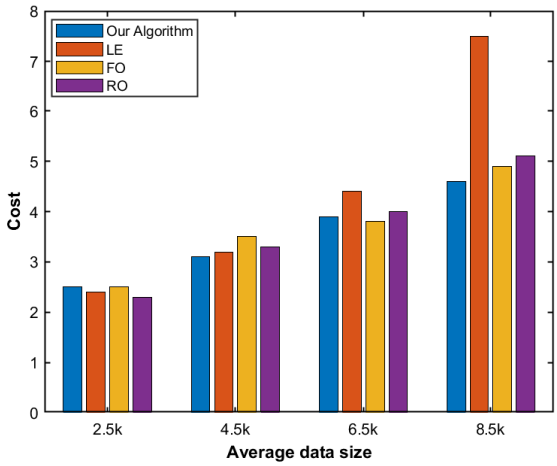

Figure 5: Impact of different sizes of computing tasks on the total system cost.

prioritises reducing the rental price. Conversely, larger values of λ_A correspond to the time-sensitive scenario, where the agent focuses on minimising the AoI cost. Notably, there is a close correlation between the trends of AoI cost and energy consumption, owing to the positive relationship between lower AoI and transmission energy consumption associated with offloading computation compared to local computing. These observations highlight the intricate relationships among AoI, energy consumption, and rental prices, underscoring the challenge of simultaneously optimising these factors.

Fig. 4 depicts the impact of the maximum available energy of the UAV on the system cost. Using Eq. (20), we calculate that the UAV's maximum propulsion energy consumption is 3209.8J, and the minimum hovering energy consumption is 421.2J. Consequently, the UAV's maximum available energy ranges from 500J to 3000J, and we compare this range to the scenario without an energy limit.

As the available energy of the UAV increases, we observe a slight decrease in the system cost. This phenomenon arises because, with higher available energy, the UAV can enhance its flight capacity and provide computing services to more vehicle users. Furthermore, the marginal decrease in system cost highlights the adaptability and tunability of our algorithm, demonstrating its capability to achieve favourable performance under varying constraints. Additionally, we notice that the AoI cost and penalty gradually decrease with the increase in available energy, while the rental price gradually increases. This trend indicates a corresponding enhancement in the UAV's service capability. When the initial energy of the UAV reaches 3000J, the penalty approaches zero, suggesting that the UAV's service capability remains largely unaffected by the energy constraint at this level.

Fig. 5 demonstrates the impact of the size of computing tasks on the total system cost. The average data size of computing tasks during each time slot generated by each vehicle gradually increases from 250KB to 850KB. As expected, both the total system cost and the AoI cost increase as the task data size increases. In Fig. 6, we present the mean delay for the three algorithms in completing high-priority tasks. This figure displays the mean delay under high traffic conditions with 45 vehicles per kilometre. The overall delay is equal to the sum of the delays for all tasks. If there are M tasks, and the delay for each task m is μ_m, then the overall delay is calculated as: Overall Delay $= \sum_{m=1}^{M} mu_m$. Also, the number of tasks and subtasks are denoted as N, representing the total count of tasks and subtasks involved in the process. The mean delay is divided by the total number of tasks and subtasks. It represents the average delay per task or subtask. Mathematically, it is expressed as:Mean Delay $= \frac{\text{Overall Delay}}{N} = \frac{\sum_{m=1}^{M} \mu_m}{N}$. The mean delay measures the average delay experienced per task or subtask by dividing the total accumulated delay by the total number of tasks and subtasks. Regarding the proposed algorithm and (FO) which is based on the queuing algorithm, the mean delay for both algorithms is comparable and rises as the maximum tolerable delay in the network increases.

Moreover, the system cost of the proposed algorithm and Queue-based strategy experiences a significant decrease as the data size increases. Notably, the proposed algorithm consistently exhibits the best performance across varying data sizes. For instance, when the average data size is small (e.g., 250KB), the proposed algorithm reduces the total system cost by 9.46%, 4.36%, and 6.99% compared to LE, FO, and RO strategies, respectively. These reductions increase to 58.76%, 28.55%, and 26.64% when the data size increases to 850KB. This result further underscores the versatility and efficacy of our algorithm in handling diverse computing tasks.

The proposed algorithm is applicable in real-world scenarios such as disaster response, traffic management, and environmental monitoring, where UAV-assisted VEC systems can enhance operational efficiency by autonomously optimizing UAV paths and task allocation. To adapt the SAC algorithm for these applications, employing lightweight model versions and edge computing optimizations can reduce processing times and energy consumption, addressing the constraints of UAV operations. For scalability in complex environments, tech-

Authorized licensed use limited to: University of Durham. Downloaded on January 20,2025 at 12:15:15 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. 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/JIOT.2025.3529836

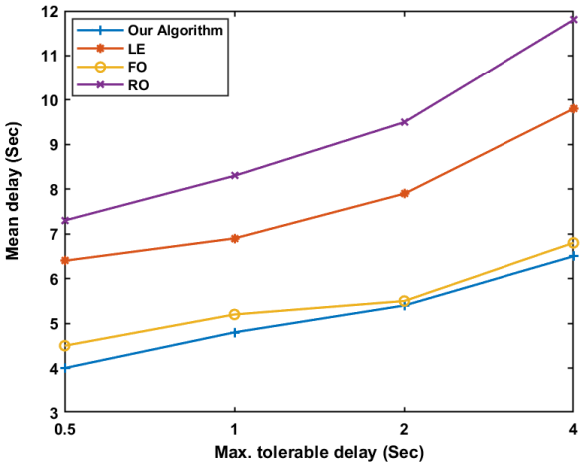

Figure 6: Mean delay as a function of the maximum tolerable delay for completing tasks, illustrating the trade-off between delay and task completion constraints.

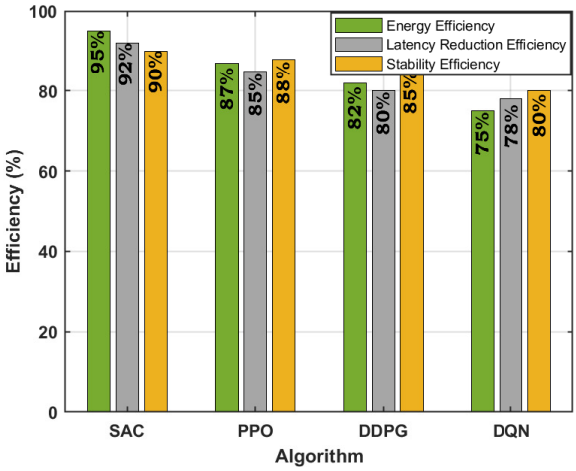

Figure 7: Comparative analysis of energy efficiency, latency reduction, and stability in UAV-assisted VEC deployment, highlighting the impact of each factor on overall system performance.

niques such as multi-agent coordination and hierarchical reinforcement learning enable efficient management of increased numbers of UAVs and diverse user demands, ensuring responsiveness even in dynamic and resource-constrained conditions

Figure 7 is a grouped bar chart comparing the performance of SAC, PPO, DDPG, and DQN algorithms across three key metrics: Energy Efficiency, Latency Reduction Efficiency, and Stability Efficiency during the deployment phase. Each algorithm is represented by a cluster of three bars, with different colours indicating the respective metrics. The chart shows that SAC outperforms the other algorithms in all three categories, achieving the highest values for energy, latency reduction, and stability. PPO follows as the next best performer, while DDPG and DQN show comparatively lower efficiency levels across these metrics. The values are displayed inside the bars, making it easy to identify the exact percentages for each metric,

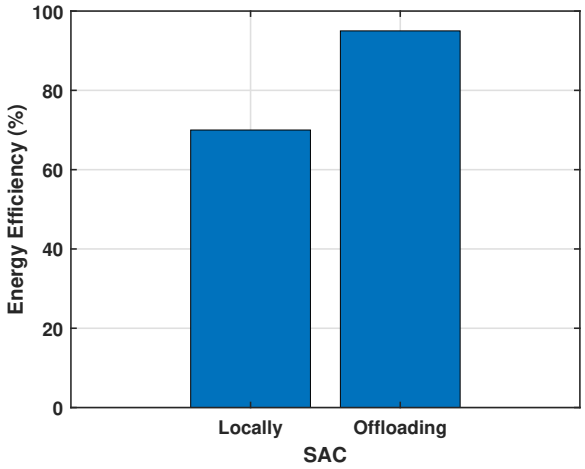

Figure 8: Energy Efficiency Comparison: SAC with and without Offloading.

with SAC having notable efficiency advantages in deployment operations.

Figure 8 demonstrates that SAC with offloading achieves substantially higher energy efficiency 95% compared to SAC without offloading 70%. This improvement is due to the more effective distribution of computational tasks, allowing for reduced onboard energy use and optimized energy consumption at the edge, thus enhancing overall system performance.

VII. Conclusion

The paper addresses the challenge of real-time and efficient processing of vehicle tasks in a UAV-assisted VEC network by optimising trajectory control and computation offloading. It proposes an optimisation scheme based on the soft actor-critic (SAC) algorithm to handle the non-convex problem, considering the stochastic nature of task arrivals and dynamic network conditions. The proposed algorithm outperforms traditional methods and four baseline strategies in terms of convergence speed and optimisation objectives. Simulation results highlight its superiority in reducing system cost and enhancing performance across various scenarios by adjusting weights on AoI cost, energy consumption, and rental price. Future research will explore the impact of packet loss in high mobility scenarios, collaborative offloading between vehicles or RSUs, and integration of new optimisation metrics and technologies like NOMA and RIS to further improve transmission and computation efficiency while ensuring reliability.

Acknowledgment

This work was supported in part by the U.K. Engineering and Physical Sciences Research Council under Grant EP/P03456X /1 and EP/X013162/1.

References

[1] F. Samie, V. Tsoutsouras, L. Bauer, S. Xydis, D. Soudris, and J. Henkel, "Computation offloading and resource allocation for low-power IoT edge devices," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 2016, pp. 7–12.

Authorized licensed use limited to: University of Durham. Downloaded on January 20,2025 at 12:15:15 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. 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/JIOT.2025.3529836

[2] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *Mobile networks and applications*, vol. 26, pp. 1145–1168, 2021.

[3] X. Peng, Z. Han, W. Xie, C. Yu, P. Zhu, J. Xiao, and J. Yang, "Deep reinforcement learning for shared offloading strategy in vehicle edge computing," *IEEE Systems Journal*, 2022.

[4] L. Yao, X. Xu, M. Bilal, and H. Wang, "Dynamic edge computation offloading for internet of vehicles with deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[5] S. Goudarzi, M. H. Anisi, H. Ahmadi, and L. Musavian, "Dynamic resource allocation model for distribution operations using SDN ," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 976–988, 2020.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[7] J. Wang, K. Liu, and J. Pan, "Online UAV-mounted edge server dispatching for Mobile-to-Mobile edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 1375–1386, 2019.

[8] P. A. Apostolopoulos, G. Fragkos, E. E. Tsiropoulou, and S. Papavassiliou, "Data offloading in uav-assisted multi-access edge computing systems under resource uncertainty," *IEEE Transactions on Mobile Computing*, vol. 22, no. 1, pp. 175–190, 2021.

[9] 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, 2018.

[10] M.-A. Messous, H. Sedjelmaci, N. Houari, and S.-M. Senouci, "Computation offloading game for an UAV network in mobile edge computing," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.

[11] F. Pervez, A. Sultana, C. Yang, and L. Zhao, "Energy and latency efficient joint communication and computation optimization in a multi-uav assisted mec network," *IEEE Transactions on Wireless Communications*, 2023.

[12] N. T. Hoa, N. C. Luong, D. Van Le, D. Niyato *et al.*, "Deep reinforcement learning for multi-hop offloading in uav-assisted edge computing," *IEEE Transactions on Vehicular Technology*, 2023.

[13] M. A. Potter and K. A. D. Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, no. 1, pp. 1–29, 2000.

[14] W. Chen, T. Weise, Z. Yang, and K. Tang, "Large-scale global optimization using cooperative coevolution with variable interaction learning," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2010, pp. 300–309.

[15] M. N. Omidvar, X. Li, and X. Yao, "Cooperative co-evolution with delta grouping for large scale non-separable function optimization," in *IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 1–8.

[16] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 378–393, 2013.

[17] Y. Mei, M. N. Omidvar, X. Li, and X. Yao, "A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization," *ACM Transactions on Mathematical Software (TOMS)*, vol. 42, no. 2, pp. 1–24, 2016.

[18] W. L. Tan, W. C. Lau, O. Yue, and T. H. Hui, "Analytical models and performance evaluation of drive-thru internet systems," *IEEE Journal on selected areas in Communications*, vol. 29, no. 1, pp. 207–222, 2010.

[19] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7944–7956, 2019.

[20] C. Qiu, Y. Hu, Y. Chen, and B. Zeng, "Deep deterministic policy gradient (ddpg)-based energy harvesting wireless communications," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8577–8588, 2019.

[21] Z. Chen and X. Wang, "Decentralized computation offloading for multi-user mobile edge computing: A deep reinforcement learning approach," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, no. 1, p. 188, 2020.

[22] L. Zhao, K. Yang, Z. Tan, H. Song, A. Al-Dubai, A. Y. Zomaya, and X. Li, "Vehicular computation offloading for industrial mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7871–7881, 2021.

[23] M. A. Ali, Y. Zeng, and A. Jamalipour, "Software-defined coexisting UAV and WiFi: Delay-oriented traffic offloading and UAV placement," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 988–998, 2020.

[24] C.-F. Liu, S. Samarakoon, M. Bennis, and H. V. Poor, "Fronthaul-aware software-defined wireless networks: Resource allocation and user scheduling," *IEEE Transactions on Wireless Communications*, vol. 17, no. 1, pp. 533–547, 2017.

[25] M. Chen, W. Saad, and C. Yin, "Liquid state machine learning for resource and cache management in LTE-U unmanned aerial vehicle (UAV) networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 3, pp. 1504–1517, 2019.

[26] N. Zhao, X. Pang, Z. Li, Y. Chen, F. Li, Z. Ding, and M.-S. Alouini, "Joint trajectory and precoding optimization for UAV-assisted NOMA networks," *IEEE Transactions on Communications*, vol. 67, no. 5, pp. 3723–3735, 2019.

[27] Y. Zhu, G. Zheng, and M. Fitch, "Secrecy rate analysis of UAV-enabled mmWave networks using matérn hardcore point processes," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 7, pp. 1397–1409, 2018.

[28] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, 2017.

[29] S. Goudarzi, S. A. Soleymani, W. Wang, and P. Xiao, "Uav-enabled mobile edge computing for resource allocation using cooperative evolutionary computation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 59, no. 5, pp. 5134–5147, 2023.

[30] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV -mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 3, pp. 2049–2063, 2017.

[31] M. N. Soorki, M. Mozaffari, W. Saad, M. H. Manshaei, and H. Saidi, "Resource allocation for machine-to-machine communications with unmanned aerial vehicles," in *2016 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2016, pp. 1–6.

[32] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2016.

[33] M. Cui, G. Zhang, Q. Wu, and D. W. K. Ng, "Robust trajectory and transmit power design for secure UAV communications," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 9, pp. 9042–9046, 2018.

[34] S. Goudarzi, M. H. Anisi, D. Ciuonzo, S. A. Soleymani, and A. Pescape, "Employing unmanned aerial vehicles for improving handoff using co-operative game theory," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 2, pp. 776–794, 2020.

[35] 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.

[36] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, 2017.

[37] K. Wang, K. Yang, and C. S. Magurawalage, "Joint energy minimization and resource allocation in C-RAN with mobile cloud," *IEEE Transactions on Cloud Computing*, vol. 6, no. 3, pp. 760–770, 2016.

[38] Z. Qin, Z. Wei, Y. Qu, F. Zhou, H. Wang, D. W. K. Ng, and C.-B. Chae, "Aoi-aware scheduling for air-ground collaborative mobile edge computing," *IEEE Transactions on Wireless Communications*, 2022.

[39] Y. Wang, Z.-Y. Ru, K. Wang, and P.-Q. Huang, "Joint deployment and task scheduling optimization for large-scale mobile users in multi-UAV-enabled mobile edge computing," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3984–3997, 2019.

[40] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

Authorized licensed use limited to: University of Durham. Downloaded on January 20,2025 at 12:15:15 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

Durham University
Durham Research Online

Citation on deposit: Goudarzi, S., Soleymani, S. A., Anisi, M. H., Jindal, A., & Xiao, P. (online). Optimizing UAV-Assisted Vehicular Edge Computing With Age of Information: A SAC-Based Solution. IEEE Internet of Things Journal, https://doi.org/10.1109/jiot.2025.3529836

For final citation and metadata, visit Durham Research Online URL: https://durham-repository.worktribe.com/output/3342504

Copyright statement: This accepted manuscript is licensed under the Creative Commons Attribution 4.0 licence.
https://creativecommons.org/licenses/by/4.0/