Intelligent Pricing Model for Task Offloading in Unmanned Aerial Vehicle Mounted Mobile Edge Computing for Vehicular Network

Asrar Ahmed Baktayan, Student, IEEE, Ibrahim Ahmed Al-Baltah, and Abdul Azim Abd Ghani

Original scientific article

Abstract—**In the fifth-generation (5G) cellular network, the Mobile Network Operator (MNO), and the Mobile Edge Computing (MEC) platform will play an important role in providing services to an increasing number of vehicles. Due to vehicle mobility and the rise of computation-intensive and delay-sensitive vehicular applications, it is challenging to achieve the rigorous latency and reliability requirements of vehicular communication. The MNO, with the MEC server mounted on an unmanned aerial vehicle (UAV), should make a profit by providing its computing services and capabilities to moving vehicles. This paper proposes the use of dynamic pricing for computation offloading in UAV-MEC for vehicles. The novelty of this paper is in how the price influences offloading demand and decides how to reduce network costs (delay and energy) while maximizing UAV operator revenue, but not the offloading benefits with the mobility of vehicles and UAV. The optimization problem is formulated as a Markov Decision Process (MDP). The MDP can be solved by the Deep Reinforcement Learning (DRL) algorithm, especially the Deep Deterministic Policy Gradient (DDPG). Extensive simulation results demonstrate that the proposed pricing model outperforms greedy by 26% and random by 51% in terms of delay. In terms of system utility, the proposed pricing model outperforms greedy only by 17%. In terms of server congestion, the proposed pricing model outperforms random by 19% and is almost the same as greedy.**

Index terms—**Computation Offloading, Dynamic Price, System Utility, Deep Reinforcement Learning (DRL), Unmanned Aerial Vehicles (UAVs), MEC.**

I. Introduction

The drones assist cellular networks to provide a quick, efficient, and cost-effective solution. The Mobile Edge Computing (MEC) server, assisted by Unmanned Aerial Vehicles (UAV), provides services to mobile users or vehicles if they are in the same cell as the UAV-MEC server [1].

Manuscript received November 3, 2021; revised February 2, 2022. Date of publication April 6, 2022. Date of current version April 6, 2022. The associate editor prof. Vladan Papić has been coordinating the review of this manuscript and approved it for publication.

A. A. Baktayan and I. A. Al-Baltah are with the Department of Information Technology, FCIT, Sana'a University, Sana'a, Yemen (e-mails: asrar@yemenmobile.com.ye, albalta2020@gmail.com).

A. A. Abd Ghani is with the Department of Software Engineering Information Systems, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Selangor, Malaysia (e-mail: azim@upm.edu.my).

Digital Object Identifier (DOI): 10.24138/jcomss-2021-0154

As long as the latency between the vehicle and the serving UAV-MEC server satisfies the service delay requirements, the UAV-MEC server that serves the vehicle cannot be too far away from the vehicle, i.e. the delay spread between them should be at a minimum.

However, due to obstructions such as dense buildings, a lack of infrastructure in some zones, and even the mm-wave Line of Sight (LOS), the quality of the connection is difficult to ensure. As a result, due to their terrain-agnostic and flexible deployment capabilities, UAVs have become one of the primary techniques for establishing communication linkages between the two ends [2].

In actuality, the UAV with MEC is unable to manage the amount of data offloaded by vehicles, and the UAV-MEC is unable to observe the vehicle's offloading profit function. Furthermore, vehicles are hesitant to report profile functions as they contain sensitive information such as battery states [3]. Given that, various vehicles place varying negative values on delay, as some are more delay-sensitive than others [4]. Thus, Mobile Network Operators (MNOs) find themselves intertwined in this profit-generating sector with competition for cached content in shared storage controlled by numerous MNOs, as well as pricing concerns [5]. Furthermore, typical optimization approaches cannot effectively optimize the computing resources and energy consumption of the UAV-assisted MEC system due to its complexity [6]. Vehicular computation offloading is a well-received approach for executing vehicles' delay-sensitive and/or computing-intensive tasks. The response time of vehicular computation offloading can be reduced by employing MEC, which has a high processing capacity and brings these computation tasks closer to the end vehicles.

Deep Reinforcement Learning (DRL), a significant extension of the Reinforcement Learning (RL) technique, has been linked to several complex online optimization problems involving enormous arrangement spaces. Traditional network optimization techniques, such as greedy linear programming and greedy search, meet the network's immediate needs, whereas RL algorithms filter the entire system by considering every possible scenario. Vehicles determine the best strategy for real-time allocation of resources for dynamic networks, such as wireless networks, where conditions change regularly [7]. DRL is a powerful sequential decision-making approach to maximize long-term framework execution in an unknowable arrangement environment, which has been widely associated with resource competition, agent interconnection, and other reasons. DRL may help improve RL proficiency by expanding the scope of the arrangement and is thus certified to make steps in 5G network efficiency [8].

Input size and computational need (CPU cycles per bit), commonly known as "complexity factors," are the most important factors in task processing. These elements are critical to explaining the various computational needs. Some operations, such as applying filters to an image, take fewer CPU cycles than applying a face detection algorithm to a video [9]. Vehicles demand resources from the edge cloud to complete computation tasks, while the edge cloud must keep its available CPU cycles for computing the total offloaded data below its calculation capacity. As a result, the edge cloud is thought to price CPU cycles to manage the demand and supply of processing resources. The vehicles then divide their input data for local computation and offloading according to the prices set by the UAV-MEC server [10].

Computation offloading is not always effective, and a poor option will result in increased processing time and energy usage. Thus, pricing is an important tool for allocating resources to users (vehicles) who value them the most and for reducing congestion in resource-constrained situations [11]. The majority of the works presented optimization techniques for static users, ignoring the random fluctuations in task production and the user's mobility. This results in a severe decline in QoS and network performance degradation. Because the UAV-MEC flies in the air and provides services to ground vehicles, it consumes a lot of energy [3]. As the UAV-MEC is flying at a fixed trajectory in the air, providing services (for example, computing and caching) to ground vehicles which are energy-consuming. A longer time helps the UAV-MEC meet more demands while leaving less energy to serve them.

To ensure the UAV-MEC's economic capability, it is critical to balance flying time and service capacity within a limited energy budget. Additionally, when the UAV-MEC is flying at a hotspot for a given period, how to dynamically price the limited UAV-MEC capacity to ground vehicles for profit maximization is another issue. This is challenging under incomplete information and vehicle randomness in arrival and their private valuations of buying UAV resources. Due to the uncertainty of both the task arrival rates and the prices of the computing resources, it has been difficult to make an optimal decision online for a long time. Since both the arrival rate of a task and the price of computing resources are uncertain, UAV-MEC has been trying to dynamically price the limited capacity to maximize expected benefits while optimizing offloading decisions and minimizing latency and energy under incomplete information on the vehicle over a long period and making the best decisions online [12].

One of the most recent works is [13]. It is shown that if the remaining hovering duration of the UAV is longer or its MEC service capacity is smaller, then the UAV-MEC server should demand a higher price. The work considered that the UAV-MEC is moving but that the user device is stationary. The energy allocated to the hovering time and service capacity in a hotspot is then optimized based on the approximate optimal pricing. Furthermore, the dual problems of end-user MEC server selection and optimal data offloading, as well as MEC server optimal price setting, are studied in [14] with multiple MEC servers and multiple end-user scenarios. The proposed framework implementation is more practicable with the help of SDN, but also because of the work handled by the task offloading of static MEC and static users. Game theory and reinforcement learning are used in the proposed approach to solve the dynamic pricing problem.

The prospect-theoretic utility function is developed in [15] by quantifying the user's risk-seeking and loss behavior of the user while considering the UAV-MEC pricing mechanism. As a result, the user's pricing and risk-aware data offloading problems are described as a distributed maximization problem of each user's expected prospect-theoretic utility function, handled as a non-cooperative game among users, with the UAV and users assumed to be stationary. The problem is solved by using submodular theory to achieve Nash equilibrium. However, a task-offloading decision strategy is proposed for the MEC Internet of Vehicles (IoV) is proposed in [16] using particle swarm optimization to choose the best offloading technique. The authors established a mathematical model to calculate the cost of computation offloading for MEC, and then particle swarm optimization was used to select the best offloading strategy. Although the authors explored moving users, the MEC server is in a fixed position. The work [17] studied the task offloading delay and the consumption of computing resources in multi-user, multi-server situations using vehicular edge computing (VEC). To improve the utility of offloading in VEC networks, the authors suggested a hybrid intelligent optimization approach based on Parthenos's genetic algorithm and heuristic rules. In [18], the authors formulated a problem to reduce the amount of battery energy consumed by task execution, task data transfer, and waiting for offloaded task outcomes on mobile devices. The results demonstrated that by utilizing sliced edge computing resources, the approach reliably reduced battery energy consumption over a wide range of task complications and task deadlines, prolonging the battery lives of mobile devices. The offloading decision and resource allocation problem were investigated in [19] as a multi-user and multi-server UAV-assisted MEC environment. To ensure the quality of service for end users, they set the weighted total cost of delay, energy consumption, and the size of discarded tasks as the optimization objective. The joint optimization problem was then formulated as a Markov decision process, and the offloading policy was optimized using the soft actor–critic (SAC) deep reinforcement learning algorithm. The results demonstrated that the offloading policy optimized by the proposed SAC-based dynamic computing offloading algorithm effectively reduces the UAV-assisted MEC system's delay, energy consumption, and task size.

A. Novelty and Contributions

This work differs from others in that it considers moving UAV-MEC and moving vehicles. The purpose of this research is to present AI-driven pricing for the data offloading approach that allows vehicles to optimally offload some of their data to a UAV-MEC server for additional processing while reducing server congestion using DRL. Therefore, the vehicle saves money (time and energy) by selecting the optimal offloading option for a price announced by UAV-MEC, and UAV-MEC earns money. This paper proposes the use of dynamic pricing to improve the quality of computation offloading. The price offered to vehicles can assist UAV operators in making better use of network resources and managing congestion issues. This is difficult in vehicular communication due to rapid mobility, causing data offloading failures or when the execution time is longer than the sojourn time [20]. Mobile networks are characterized by their mobility, which is the main cause of the rapidly changing network topology and intermittent communication connectivity [21].

We investigate intelligent pricing in the UAV-MEC network with *M* vehicles and *one* mobile UAV-MEC server. There are some computational tasks in the vehicle, and some of them can be offloaded to UAV-MEC at the expense of fees. The proposed solution examines three important key metrics: latency, energy, and revenue. We formulate the problem as an MDP (Markov Decision Process).

The proposed model aims to maximize the total profit of the UAV operator, minimize latency and power consumption, and prevent resource congestion on the UAV-MEC server. Pricing strategies depend on the strategy of all offloading decisions in which the vehicle is selected. Therefore, vehicles indirectly influence each other's decisions. Using DRL, UAV-MECs can learn appropriate pricing strategies that maximize the social welfare (system utility) of UAV operators without prior knowledge of vehicle information. The contributions to this article are as follows:

1) We propose a task offloading strategy in a UAV-MEC-based vehicle network environment that considers a scenario in which both the UAV-MEC server and the vehicle move during the offloading process of a large-scale task. Influential factors considered in the calculated offload strategy include vehicle speed, coverage area, data transfer rate, and UAV-MEC congestion rate.

2) formulating the dynamic pricing strategy, which is non-convex as MDP to minimize processing delay and energy, then maximizing the system utility when the offloading operation is completed.

3) Considering the MDP model, the complexity of the state of the system is very high. In addition, dynamic pricing requires continuous action space support. Therefore, we propose a Deep Deterministic Policy Gradient (DDPG)-based computational offload approach. We adopted the DDPG algorithm to obtain the optimal pricing policy to control task offloading. Therefore, the proposed UAV network supported by MEC minimizes delay, energy consumption, and social welfare (system utility).

4) The proposed DDPG algorithm is written in Python 3.9 and runs on the TensorFlow platform. The results of the simulation using various system parameters show the efficiency of the proposed dynamic offload pricing algorithm. Compared with state of art and baseline methods, specifically, greedy and random algorithms, the proposed algorithm can adapt to a highly dynamic environment with outstanding performance.

The rest of this article is structured as follows. Section II introduces the system models; we propose a communication model, a computation model, and a pricing model for the proposed pricing algorithm. In Section III, we describe the problem formulation, joint metric, and transformation of the problem into MDP. Section IV presents the solution using the DDPG algorithm. The performance analysis is presented in Section V before the article is concluded in Section VI.

II. System Model

The proposed system consists of one UAV-mounted MEC server services vehicles. As shown in Fig. 1, the Ground Base Station (GBS) is assumed to be already computationally overloaded, or if there is no GBS, therefore unable to provide computing services for vehicle requests in the UAV-MEC hotspot area. So, at each time slot, the UAV will move in a fixed trajectory and establish communication with vehicles in TDMA (Time Division Multiplexing Access) mode. After offloading some of the vehicle's computer tasks to the UAV-MEC server, the vehicle performs the rest of the tasks locally [22]. We consider that UAV-MECs have an energy supply through solar energy [23], wind [24], or wireless power transfer (WPT) technology [25], which are considered expensive compared to the direct power supply of the ground MEC server; then we will take the energy consumption into account.

A. Communication Model

Since UAV-MEC provides computing services to all vehicles in a TDMA mode [26], we discretize the finite flying time *T* into *N* equal time slots, that is, $T = N\,\delta t$, where δt is the time interval that allows UAV-MEC to be in a certain place at a certain time slot. In the 3D Cartesian coordinate system, the UAV continues to fly at a fixed altitude *H*, where the UAV has a start coordinate [26] [27]:

$q(i) = [x(i), y(i)]^{\pi} \in R^{2\times 1}$ and the end coordinate at the time slot $i \in \{1, 2, ..., I\}$ .The coordinates of vehicles $m \in \{1, 2, ..., K\}$ is $p_m[x_m(i), y_m(i)]^{\pi} \in R^{2\times 1}$.

The channel gain of the line of sight link (LoS) between the UAV-MEC and the vehicle m can be expressed as [28]:

$$g_m(i) = \alpha_0 d_m^2(i) = \frac{\alpha_0}{\| q(i+1) - p_m(i) \|^2 + H^2} \tag{1}$$

where a α_0 denotes the channel power gain at a reference distance *d* = 1 m, and $d_m(i)$ represents the Euclidean distance between the UAV-MEC and vehicle *m*. Due to barrier blocking, the wireless transmission rate can be stated as [29]:

$$r_m(i) = B \log_2\left(1 + \frac{p_{vp}\, g_m(i)}{\sigma^2 + p_m(i) p_{NLOS}}\right) \tag{2}$$

where *B* denotes the communication bandwidth, p_{vp} denotes the transmit power of the vehicle in the upload link, σ^2 denotes the noise power, p_{NLOS} denotes the transmission loss, and $p_m(i)$ signifies the indicator of whether there is any block between UAV and vehicle m in time slot i (that is, 0 means no blockage and 1 means there is a blocking object) [26].

A. Computation Model

When the task data size or subtask is small and the vehicle computing and energy resources can handle task processing, the task execution can be calculated locally, which consumes local energy and computing resources, and causes an operation delay. Let $R_m(i) \in [0,1]$ denotes the ratio of tasks offloaded to the

UAV-MEC server, and $(1-R_m(i))$ denotes the remaining tasks that are executed locally in the vehicle.

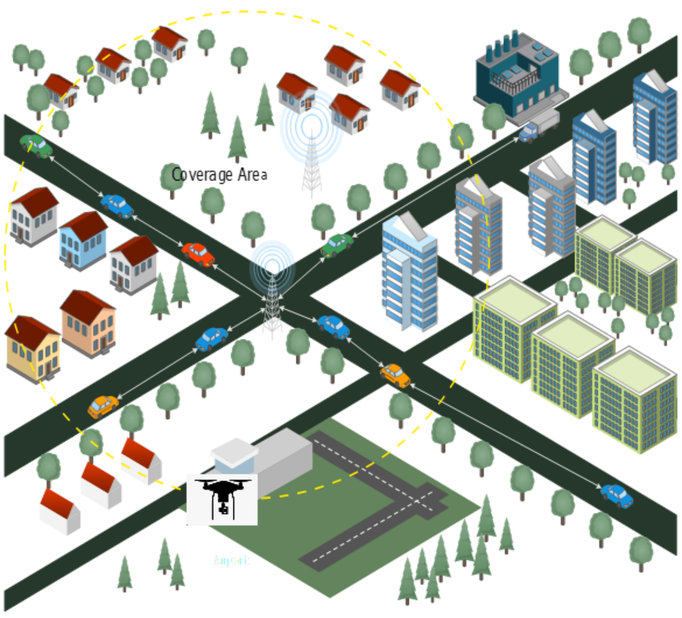

Fig . 1. UAV mounted by MEC for the vehicular network

TABLE I
NOTATIONS

Notation	Description
M	Number of vehicles
D_m	Total input data size needed for the execution of each the vehicle m
E_m	Local energy consumption
T_m	local running delay
$t_{UAV,m}$	The time for completing the task in UAV
$E_{UAV,m}$	energy consumption for execution data in UAV-MEC server
f_m	the vehicle equipment operation frequency
S_m	the required CPU cycles to accomplish one bit
φ	the UAV's price per cycle
f_{UAV}	the computation frequency of UAV-MEC server CPU
$R_m(i)$	offloading ratio
$\beta(i)$	Flight angle
$V(i)$	Speed of UAV
t_{fly}	the fixed flight time
α_0	the channel power gain at a reference distance d = 1 m
E_{fly}	The energy consumed by the flight
t_{tr}	the transmission delay
p_{UAV}	the power consumption in UAV-MEC server for execution task
t_{off}	Time for offloading to UAV-MEC
RV_{UAV}	UAV-MEC server revenue
$cong_{UAV}$	UAV-MEC server congestion ratio
U_{UAV}	System utility (social welfare)

The local running delay (T_m) is given as follows:

$$T_m = \frac{(1-R_m(i))l_m(i)s_m}{f_m} \tag{3}$$

where s_m indicates the CPU cycles required to process one bit, and f_m represents the vehicle's CPU frequency rate. At the i-th time slot, the UAV flies from position $q(i)$ to the new position:

$$q(i+1)=[x(i),v(i)t_{fly}\cos\beta(i),y(i)+v(i)t_{fly}\sin\beta(i)]^{\tau} \tag{4}$$

At a speed $v(i)\in[0,v_{\max}]$ and angle $\beta(i)\in[0,2\pi]$. The energy consumed by the flight can be expressed as

$$E_{fly}(i)=\varphi\left\|v_{UAV}(i)\right\|^2 \tag{5}$$

where $\varphi=0.5N_{UAV}t_{fly}$ [30], N_{UAV} is relevant to mass of UAV including the payload, t_{fly} is a fixed flight time. The size of the computing results delivered by the server in the UAV-MEC system is usually very small and inconsequential. As a result, the transmitting delay of the downlink is not taken into account here [30] [31] [15] [32]. The processing delay of the UAV-MEC server can be divided into two parts. One part is the transmission delay, which can be given as

$$t_{tr,m}(i)=\frac{R_m(i)l_m(i)}{r_m(i)} \tag{6}$$

The other part is the delay resulting from the computation on the UAV-MEC server, which can be given as [33]:

$$t_{UAV,m}(i)=\frac{R_m(i)l_m s_m}{f_{UAV}} \tag{7}$$

where f_{UAV} signifies the server CPU computation frequency. Similarly, the energy used to offload computing tasks to the server in the time slot i can be separated into two parts: transmission and processing. When the computation is done on the UAV-MEC server, the power consumption is calculated as follows.

$$p_{UAV,m}(i)=mf_{UAV}^3 \tag{8}$$

Then, the energy consumption of the UAV-MEC server at time slot i is given as [26]:

$$E_{UAV,m}(i)=p_{UAV,m}(i)=f_{UAV}^2 R_m(i)l_m(i)s_m \tag{9}$$

The offloading time $t_{UAV,m}(i)$ of a vehicle m comprises two parts: the uplink transmission time $t_{tr,m}(i)$ the execution time at the UAV-MEC server $t_{UAV,m}(i)$. Thus, the offloading time $t_{tr,m}(i)$ is [10]:

$$t_{off,m}(i)=t_{UAV,m}(i)+t_{tr,m}(i) \tag{10}$$

Since local computation and offloading can be performed concurrently, the required vehicle time m to execute the total l_m bits data can be expressed as [34] [29]:

$$t_{total}(i) = \max\{t_{off,m}(i), t_m(i)\} \tag{11}$$

To compute the energy consumption E_m (in Joule) of vehicle m when the task is executed locally, according to [32] [35]

$$E_m(i) = k_m s_m (f_m)^2 \tag{12}$$

where κ_m is a coefficient related to the hardware architecture of the chip. According to the measurements in [36] we set $k_m = 5\times10^{-27}$. T_m and E_m are calculated in advance since they depend on the specific attributes of vehicle m and the underlying application.

C. *UAV Offloading Price Model*

When the vehicle cannot perform task computing, the task can be offloaded to the UAV-MEC server. In this model, the UAV can function as a MEC server. This section covers the UAV's role as a MEC server, which provides vehicle computing offloading services. The delay and energy components of UAV-MEC offloading are also separated [31].

We assume that UAV revenue can be expressed as a function of the sum of tasks offloaded, and we denote the CPU cycle prices for vehicles as a set φ to the independent variable of the revenue function. Accordingly, UAV revenue can be formulated as

$$RV_{UAV}(i) = \sum_{i=1}^{T}\sum_{m=1}^{M} R_m l_m \varphi \tag{13}$$

The server will become congested if the randomly generated tasks exceed the capacity of the UAV-MEC server. Therefore, congestion control is required to ensure the stability of the UAV-MEC system. The UAV-MEC congestion ratio of UAV-MEC can be expressed as [14]:

$$cong_{UAV} = R_m(i)\frac{l_m s_m}{f_{UAV}} \tag{14}$$

III. Problem Formulation of Dynamic Pricing Algorithm for Offloading

The goal of the UAV-MEC system is to maximize revenue while minimizing delay and energy. The UAV operator wants to make the most money by selling computational capabilities to vehicles. The purpose of this paper is to come up with a pricing strategy that will maximize UAV-long-term MEC's revenue. This is complicated by the fact that the UAV-MEC server is unaware of the vehicle's personal demand information, such as job arrival rate patterns [37], movement speed, and vehicle direction. When UAV-MEC alters the price choice in a multi-objective problem, the agents' environment is modified. As a result, each vehicle can only get the price of the UAV-MEC and not the offloading choice of other vehicles [29]. Furthermore, task execution delay and system utility are significant performance measures for autonomous driving. Long delay and low system utility will affect the quality of experience (QoE) of the vehicle.

A. *Joint Performance Metrics*

The latency, energy usage, and revenue measurements of the UAV-MEC network provide three-dimensional performance indicators. You may reduce total latency, save energy, increase total income, and improve communication and processing performance by using UAV-MEC. Thanks to the ongoing development of wireless mobile communication technologies, large-capacity data transfer is no longer a restriction on wireless networks. Therefore, the vehicle can offload more tasks to the UAV-MEC server, which has more computing capability, and the total delay time is reduced. According to Equation (13), the vehicle, on the other hand, will have to pay more to the UAV-MEC to offload more data. The key to improving vehicle performance, as described above, is to apply price control to develop a good offloading ratio. We strive to reduce latency and energy consumption while maximizing UAV-MEC's benefits. According to the description, it is a multi-objective optimization problem to improve vehicle performance (lower latency and energy consumption) while providing benefits to the UAV operator. In addition, vehicles may face an urgent task or tend to pay less in different scenarios [29]. We can assess system performance from three-dimensional viewpoints, but this poses a significant challenge in terms of system optimization. We examine a weighted performance factor based on delay, energy consumption, and price to simplify the measurement of system performance and the associated optimization [16]. The weighted factor λ turns the utility of the system of multi-objective optimization problem system utility -or called social welfare- [38][39] into a linearly weighted objective function [40]. The formulated problem can be given by:

$$\max_{\{\varphi_i, q(i+1)\}} U_{UAV}(i) = \sum_{i=1}^{I}\sum_{m=1}^{M} \lambda1(E_m(i) + E_{UAV}(i)) + \lambda2(t_{total}(i) + (1-\lambda1-\lambda2)RV_{UAV}(i) \tag{15}$$

s.t

C1: $$\sum_{i=1}^{I}\sum_{m=1}^{M} R_m(i) l_m(i) = d_m \quad \forall_m \in M \tag{15a}$$

C2: $$p_m \in [0,1] \quad \forall i, m \tag{15b}$$

C3: $$0 \le R_m(i) \le 1 \tag{15c}$$

C4: $$\sum_{i=1}^{I} R_m(i) l_m(i) s \le f_{UAV} \tag{15d}$$

C5: $$q(i) \in \{(x(i), y(i)) \mid x(i) \in [0,L], y \in [0,W]\}, \ \forall_i \tag{15g}$$

C6: $$p(i) \in \{(x_m(i), y_m(i)) \mid x_m(i) \in [0,L], y_m \in [0,W] \tag{15h}$$

C7: $$\sum_{i=1}^{I}(E_{fly,m}(i)+E_{UAV,m}(i)\le E_{battary},\ \forall_M \tag{15i}$$

C8: $$0\le \lambda 1\le 1,\ 0\le \lambda 2\le 1,\ 0\le \lambda 1+\lambda 2\le 1 \tag{15j}$$

C1 ensures that the summation of the subtasks assigned to UAV-MEC is equal to the vehicle data that needs to be offloaded. *C2* is the blockage probability. Constraint *C3* ensures that the total partition of vehicle offloading decisions to UAVs should be 1, and *C4* ensures that the number of CPU cycles required to perform vehicle tasks cannot exceed the UAV-MEC capability. *C5* and *C6* show that vehicles and UAV can only move in the given area. Finally, constraint *C7* ensures that the flying and computational energy consumption cannot exceed the battery energy. Finally, the constraint *C8* ensure that the value of $\lambda 1$ and $\lambda 2$ is between zero and one, and the summation of $\lambda 1,\ \lambda 2$ is between zero and one.

In the formulated problem, each offloading decision has an impact on the state of the system, and the channel environment changes dynamically when the next offloading decision is made. As a result, this optimization problem qualifies as a sequential choice problem. $\lambda 1, \lambda 2$, and $(1-\lambda 1-\lambda 2)$ denote the weight factors of the latency, energy consumption, and price of the overall utility of the system. The $U_{UAV}(i)$ is the system utility for completing the m-th vehicle computational tasks. The weighted factor determines the relationship between delay, energy, and price, thus affecting the decision to optimize the system resources [41]. The use of a weighted factor simplifies a multi-objective optimization problem (NP-hard) into a single-objective optimization problem and allows equation (15) to be applied to additional scenarios. When *λ1* becomes larger in a particle, energy consumption becomes more essential in the overall utility; when *λ2* increases, the latency becomes more important in the system utility; when the value of *(1 − λ1 − λ2)* becomes larger, the price plays a more important role in the overall social welfare [41].

B. Deep Reinforcement Learning Formulation

Vehicles are unwilling to expose their private information in certain situations. Therefore, a DRL approach is employed to allow UAV-MEC to learn the optimal pricing strategies directly from the negotiation history without prior knowledge about any vehicle. To address vehicle resource constraints and vehicle QoE requirements, we have formulated an optimization problem. In this section, we have explored DRL technology to achieve the optimal solution. As a result, backhaul network congestion can be minimized, which is directly related to the resource utility, QoE, energy utilization, and profit of the UAV-MEC server. However, every vehicle is potentially selfish, meaning that they try to capitalize on their utility through monopolizing the available computing within their operational environment.

The proposed model integrates deep learning perceptual capability, RL optimal decision-making ability, and the adaptive pricing algorithm for UAV-MEC networks [42]. Indeed, due to the lack of prior information, it is impractical to solve these time-related optimization problems. Even with prior information, this optimization problem is difficult to solve because it is non-convex, the revenue is a function of the price, and delays are affected by the distance traveled. Still, thanks to the rapid development of RL, it is possible to apply the DRL algorithm to interact with the time-varying UAV-MEC environment.

C. DRL for Dynamic Pricing Algorithm For Offloading

We formulate the dynamic offloading pricing problem as an MDP. Then, we present the DRL approach for UAV-MEC to learn the optimal pricing strategies. Furthermore, UAV-MEC acts as an agent in MDP, and the environment consists of *M* vehicles.

We define the stochastic pricing strategy of UAV-MEC as MDP. The MDP can be defined as a tuple *S, A, P, R,* where S is the state space, *A* is the action space, *P* is the transition probability from the current state S to the next S+1 after the action *A* is executed, and $R\to S\times A$ is instantaneous/ immediate reward function. We denote this $\pi: S\to P(A)$ as a ‘‘policy’’ which is a mapping from state to action. MDP aims to find an optimal policy, which can maximize the expected accumulated rewards.

$$R_i=\sum_{l=i}^{l-i}\gamma^{l-i}r_i \tag{16}$$

where $\gamma\in[0,1]$ is the discount factor, $r_l=r(s_l,a_l)$ is the immediate reward at l-th time slot l. Under policy π, the expected discounted return from the state s_i is defined as a state value function, that is

$$V_\pi(s_i)=E_\pi[R_i\mid S_i] \tag{17}$$

Similarly, the expected discounted return after taking action a_i in state s_i under a policy p_i is defined as an action value function, that is

$$Q_\pi(s_i)=E_\pi[R_i\mid s_i,a_i] \tag{18}$$

The recurrent relationship of the state value function and the action value function can be represented respectively, using the Bellman equation,

$$V_\pi(s_i)=E_m[r(s_i,a_i)+\gamma V_\pi(s_{i+1})] \tag{19}$$
$$Q_\pi(s_i)=E_m[r(s_i,a_i)+\gamma Q_\pi(s_{i+1},a_{i+1})] \tag{20}$$

Because we want to identify the best policy, we may use the optimal value function to find the best action in each state. The optimal state value function is defined as follows:

$$V_*(s_i)=\max_{a_i\in A}E_\pi[r(s_i,a_i)+\gamma V_\pi(s_{i+1}) \tag{21}$$

The optimal action value function tracks the optimal policy π_*, we can write V_* in terms of Q_* as
follows:

$$Q_*(s_i,a_i)=E_\pi r(s_i,a_i)+\gamma V_\pi(s_{i+1}) \tag{22}$$

IV. DDPG BASED PRICING FOR COMPUTATION OFFLOADING OPTIMIZATION

To solve equation (15), we use the standard actor-critic framework based on the policy gradient approach [43]. More precisely, for UAV-MEC, the actor network is $\mu(s \mid \theta^{\mu})$, which generates a stochastic pricing policy μ while the critic network is $Q(s,a \mid \theta^{Q})$ .In addition, as shown in Fig. 2 both the actor network and the critic network contain a target network with the same structure: actor target network μ' with parameter $\theta^{\mu'}$, critic target network Q' with parameter $\theta^{Q'}$:

$$L(\theta^{Q}) = E_{\mu'}[(y_i - Q(s_i, a_i \mid \theta^{Q}))^2] \tag{23}$$

where $y_i = r_i + \gamma Q(s_{i+1}) \mid \theta^{Q}$, the policy gradient can be updated by the chain rule:

$$\nabla_{\theta^{\mu}} J \approx E_{\mu'}[(\nabla_{\theta^{\mu}} Q(s,a \mid \theta^{Q}))\mid_{s=s_I, a=\mu(s\mid\theta^{\mu})}] =$$
$$E_{\mu'}[(\nabla_{q} Q(s,a \mid \theta^{Q}))\mid_{s=s_I, a=\mu(s\mid\theta^{\mu})} E_{\mu'}[(\nabla_{\theta^{\mu}} Q(s,a \mid \theta^{Q}))\mid_{s=s_i}] \tag{24}$$

A. MDP Formation

In the MEC system assisted by UAVs, *M* vehicles, a UAV-MEC server, and their environment jointly determine the state space as shown in Fig. 2. The states, actions, and rewards of the system at time slot *i* can be defined as

A.1 *State* Space

We denote the states of UAV_MEC at the *i*-th time slot as:

$$s_i = E_{battery}(i) q(i), p_1(i), \ldots, p_M(i), \\ D_{remain}(i), D_1(i), \ldots, D_M(i), k_1(i), \ldots, k_M(i) \tag{25}$$

where $E_{battery}(i)$ denotes the remaining energy of the UAV battery at *i-th* time slot, $q(i)$, denotes the UAV location information, $p_M(i)$, denotes the location information of vehicle m, served by the UAV-MEC. $D_{remain}(i)$ denotes the size of the remaining tasks that the system needs to complete in the whole period, $D_M(i)$ denotes the total task size randomly generated by vehicle m in the i-th slot, and $k_m(i)$ denotes an indication of whether the signal of vehicle m is blocked by obstacles. Especially when *i = 1*, $E_{battery}(i) = E_b$ and $D_{remain}(i) = D_m$. Therefore, the state of the UAV is comprised of the records between UAV-MEC and vehicle m. We call it a 'history matrix'.

A.2 Action Space

Based on the current state of the system and the observed environment as shown in Fig. 2, the agent selects actions, including the price announced by UAV-MEC $\varphi(i)$, offloading ratio to UAV-MEC $R_m(i)$, the flight speed of UAV $v(i)$, and the weighted factor $\lambda 1(i)$, $\lambda 2(i)$. The action a$_i$ can be denoted as:

$$a_i = (\varphi(i), R_m(i), V(i), \lambda 1(i), \lambda 2(i)) \tag{26}$$

It is worth noting that the actor network in DDPG outputs is continuous actions. The flight speed of the UAV can be accurately optimized in a continuous action space similar to [26]. On the other hand, the UAV pricing strategy profile of UAV at *i-th* time-slot in which φ is defined as a continuous variable.

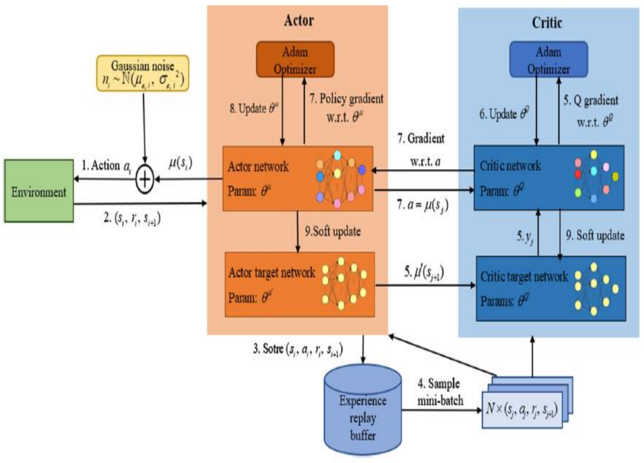
Actor
Critic
Adam Optimizer
Adam Optimizer
Gaussian noise $n_i \sim N(\mu_{e,i}, \sigma_{e,i}^2)$
8. Update θ^{μ}
7. Policy gradient w.r.t. θ^{μ}
6. Update θ^{Q}
5. Q gradient w.r.t. θ^{Q}
7. Gradient w.r.t. a
1. Action a_i
$\mu(s_i)$
Environment
Actor network Param: θ^{μ}
Critic network Param: θ^{Q}
2. (s_i, r_i, s_{i+1})
$7. a = \mu(s_j)$
9.Soft update
5. y_j
9. Soft update
Actor target network Param: $\theta^{\mu'}$
5. $\mu'(s_{j+1})$
Critic target network Params: $\theta^{Q'}$
3. Sotre (s_i, a_i, r_i, s_{i+1})
Experience replay buffer
4. Sample mini-batch
$N \times (s_j, a_j, r_j, s_{j+1})$

Fig. 2. The architecture of the DDPG learning [26]

A.3 Reward Function

We aim to maximize the reward by balancing the processing delay, energy consumption, and the profit of the UAV-MEC server. On the other hand, the reward aims to maximize the social welfare (system utility) as defined in equation (15), as follows:

$$r_i = (s_i, a_i) = U_{UAV}(i) \tag{27}$$

the long-term average reward of the system can be expressed using the Bellman equation as follows.

$$Q_{\mu}(s_i, a_i) = E_{\mu}[r(s_i, a_i) + \gamma Q_{\mu}(s_{i+1}, \mu(s_{i+1}))] \tag{28}$$

The scaling factor in the proposed approach is the difference between the highest and minimum values of each variable in the state space. The proposed state normalization algorithm can handle the problem of input variable magnitude differences [44] [26].

<u>Algorithm 1: Dynamic Offloading Pricing Algorithm for the UAV-MEC network</u>

```
Input:
Training episode length E, training sample length I;
Cretic learning rate $\alpha_{cretic}$, actor learning rate $\alpha_{actor}$;
Discount factor $\gamma$, soft update factor $\tau$ ;
```

```
Experience replay buffer Bm, mini-batch size N;
Gaussian-distributed behavior noise n with the mean value $\mu_e = n_0$ and standard deviation $\sigma_{e,i}=\sigma_e$
Current state:
$E_{battery}(i) q(i), p_1(i), ...., p_M(i),$
$D_{remain}(i), D_1(i), ..., D_M(i), k_1(i), ..., k_M(i)$
Scaling factor: $\gamma_b$, $\gamma_x$, $\gamma_y$, $\gamma_{Drm}$, $\gamma_v$
Output: reward ri
1-  Randomly initialize the weight of actor network $\theta^{\mu}$ and the cretic network $\theta^{Q}$, respectively.
2-  Initialize the target network with weight $\theta^{\mu} \leftarrow \theta^{\mu'}$ and $\theta^{Q} \leftarrow \theta^{Q'}$ and replay buffer $B_m$.
3-  For each episode e=1,2, …., E do
4-  Reset the simulation parameters of the UAV-MEC network and obtain the preliminary observation state $s_i$
5-  For i = 1,2,..., I do
6-  Normalize $s_i$ to $s_i'$ [26] :
      $E_{battery}(i)' = \frac{E_{battery}}{\gamma_b}, x(i)' = \frac{x(i)}{\gamma_x}, y(i)' = \frac{y(i)}{\gamma_y}, x_m(i)' = \frac{x_m(i)}{\gamma_x},$
      $y_m(i)' = \frac{y_m(i)}{\gamma_y}, K_m(i)' = \frac{K_m(i)}{\gamma_v}, D_{remain}(i)' = \frac{D_{remain}(i)}{\gamma_{Drm}}$
7-  Get the action with actor network $\theta^{\mu}$ and behavior noise ni;
      $a_i = \min(\max(\mu(s_i' \mid \theta^{\mu}) + n_i - 1), 1)$
8-  Perform action ai, and get the reward ri from Equation (28) and observe the next state $s_{i+1}$
9-  Normalize $s_{i+1}$ to $s'_{i+1}$
10- If the replay buffer is not full, then
11- Store transition $(s_i', a_i, r_i, s_{i+1}')$ in replay buffer $B_m$
12- Else
13- Randomly replace a transition in replay buffer $B_m$ with $(s_i', a_i, r_i, s_{i+1}')$
14- Randomly sample a mini-batch of I transition
      $(s_j', a_j, r_j, s_{j+1}') \ \forall j = 1, 2, 3, ..., I$ from $B_m$.
15- Set $y_j = r_j + \gamma Q'(s_{j+1}, \mu'(s_{j+1} \mid \theta^{\mu}), \theta^{Q'}$
16- Update the $\theta^{Q}$ in the cretic network by minimizing the loss:
      $L(\theta^{Q}) = \frac{1}{N}[\sum_{j=1}^{N}(y_j - Q(s_j, a_j \mid \theta^{Q}))]^2$
17- Update actor network $\theta^{\mu}$ by sampled policy gradient from equation (24)[45]
18- Soft update the target network according to [46] $\theta^{\mu'} \leftarrow \tau\theta^{\mu'} + (1-\tau)\theta^{\mu}$
    and update the actor target network according to [47]
      $\theta^{Q'} \leftarrow \tau\theta^{Q'} + (1-\tau)\theta^{Q}$
19- Perform the action ai and obtain the reward ri from Equation (28)
20- End if
21- End for
22- End for
```

V. Performance Evaluation

We perform extensive simulations to evaluate the impact of the main parameters on the performance of the dynamic pricing algorithm. Performance metrics are demonstrated as follows:
1) Total delay: The execution time of the dynamic pricing of the offloading algorithm is the time to transmit the offloading tasks plus the computation time consumed on the UAV-MEC server or the local delay.

2) System Utility (Social Welfare): The total amount of benefit received by each UAV-MEC server. In the offloading period, it aims to schedule tasks to maximize the utility of the UAV-MEC server by minimizing the delay and energy consumption cost. The weighted factors *λ1, λ2* are used to balance the three key metrics: delay, energy, and price, to obtain the optimal utility of the system.

A. Experimental Setup

There are many simulation tools that implement DRL, such as SimuLTE with Keras, integration with the OMNeT++ or MATLAB environment by using the Tensorflow C++ or Python frontend [48]. However, for Machine Learning (ML), Python is more preferred by data scientists because of the wide selection of available libraries depending on the task. Python supports important ML frameworks like TensorFlow, Keras, Theano, etc.
This work used the Python3.9 code on the Spyder 5.05 IDE (integrated development environment) on the Anaconda platform with TensorFlow 2.5.0 (CPU version) to evaluate the performance of the intelligent pricing offloading model.
Detailed simulation parameters are listed in Table II unless otherwise specified. The scaling factors in the proposed state normalization algorithm are set to

$$\gamma_b = 5\times 105, \gamma_{D_m} = 1.05\times 10^8, \gamma_x, \gamma_y = 100, \quad \text{and} \quad \gamma_v = 2.62\times 10^6$$

respectively as [26].

TABLE II
SIMULATION AND TRAINING PARAMETERS

Parameter	Default value	Parameter	Default value
M (number of vehicles)	8	Long, Width, High	100m
m UAV(mass of UAV)	9.65 kg	*T* (time)	320 s
I(iteration number)	40	v_{max}(max UAV speed)	50 m/s
t_{fly}(flying time)	1 s	a_0(channel gain)	- 30 dB
B (bandwidth)	1 MHz	σ^2 (transmission loss-LOS)	- 100 dBm
PN_{LOS}(transmission loss)	20 dB	P_{up}(Uplink transmission power)	0.1 W
E_b(UAV battery power)	500 kJ	*S* (cycle per bit	1000 cycles/bit
f_m(vehicle CPU frequency)	0.6 GHz	f_{UAV}(UAV CPU frequency	1.2 GHz
γ_b(scaling factor for remaining battery)	5×105	γ_x,γ_y(scaling factor UAV location)	100
γ_{Dm}(scaling factor for remaining sum task size)	1.05 ×108	γ_v(scaling factor for all vehicle location)	2.62 × 106
LR_A (learning rate for actor)	0.001	GAMMA (optimal reward discount)	0.001
LR_C (learning rate for critic)	0.002	BATCH_SIZE	64
var (control exploration)	0.01	MAX_EPISODES	1000

B. *Comparison with Benchmark*

The convergence performance of the intelligent pricing model is compared state of art [26] when the price to control the offloaded data is added.

Fig. 3 shows that combining pricing with the offloading ratio as modifiable factors minimizes the delay compared to using the offloading ratio alone. It is also worth noting that pricing has no effect in a local-only scenario. The price determines how much data is offloaded; if the data is always offloaded, the price has no influence because the only-offloading scenario has the same latency regardless of price. The results for the DDPG algorithm (intelligent pricing model) indicate that adding pricing to control data offloading has good convergence properties in a dynamic offloading situation.

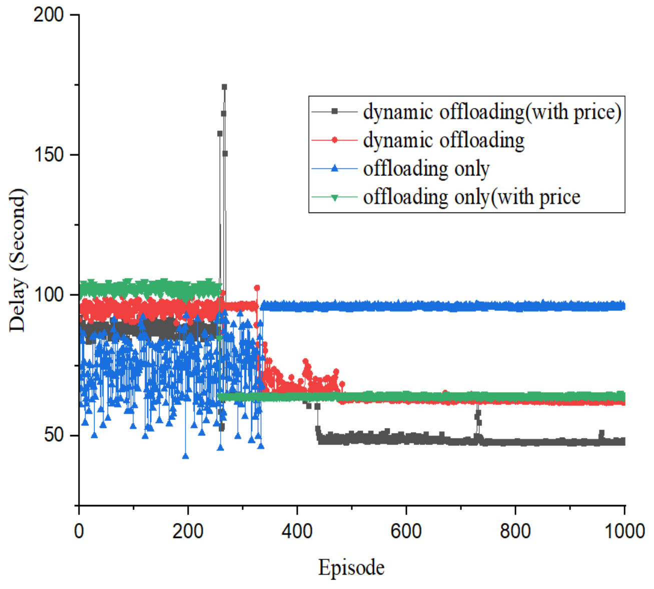

Fig. 3. Delay of offloading with and without price

C. *Convergence Performance*

Under the assignment strategy, the difference between the total revenue of completed tasks and the overall execution expenses is the system utility (social welfare). Because the revenues of the UAV-MEC and the cost of the vehicle by local execution are socially balanced.

As the number of vehicles in the system grows, the utility of the system decreases, as seen in Fig. 4. Because each time slot only has one vehicle, the other vehicle will choose local execution or wait for the next time slot. The decline in social welfare is due to the increased time delay and energy consumption costs caused by local execution. As a result of the delay and energy consumption cost, the utility of the system is reduced. The utility of the system ultimately stabilizes at a specific value after hundreds of episodes.

The delay and energy-weighted factors are minimal when the number of vehicles is low. This is because the UAV-MEC can execute most offloaded tasks within the expected timeframe, and the vehicles try to offload more data to the UAV-MEC, which increases revenue and, in return, increases the system utility. However, as the number of vehicles grows, so does the time delay and energy expense, too. This is due to the fact that the average execution time of tasks increases as more vehicles offload tasks to the UAV-MEC. On the other hand, the higher the number of vehicles, the higher the price to handle them all. It is noteworthy that at the beginning of the training, the reward is zero for more than 250. That is because of the conditions in Equation (15).

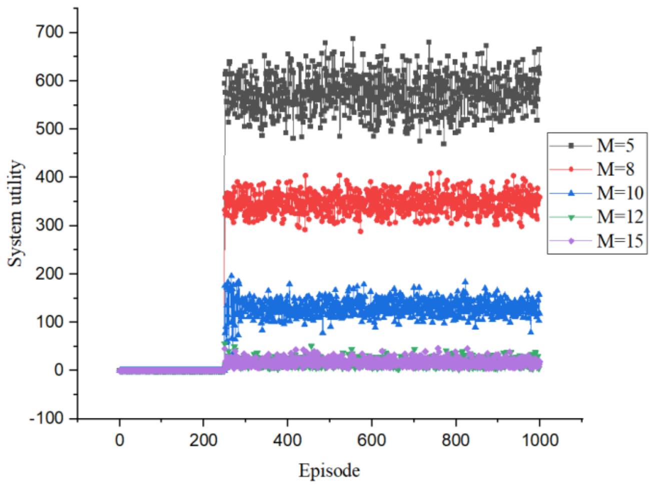

Fig. 4. System utility (Social Welfare) with different vehicle numbers

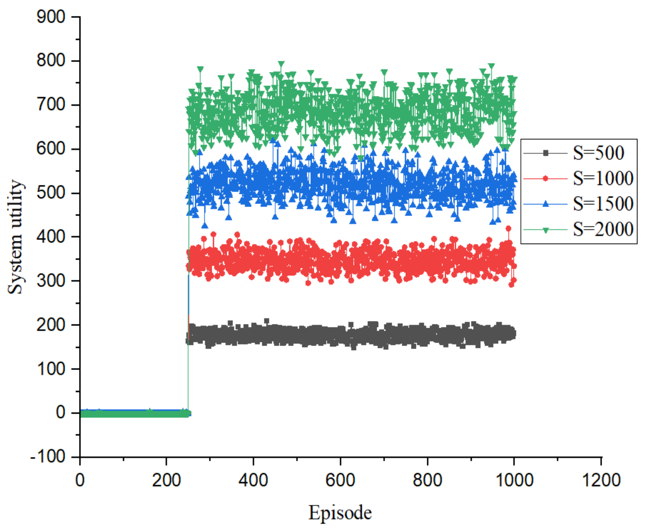

Fig. 5. System utility with different CPU cycles per bit

As seen in Fig. 5, the system utility increases as the necessary cycle to process one bit of vehicle data increases. The offloading data cycles per bit state how quickly the data should be processed and how much processing resources should be set aside for this task. As data requires more processing capability in UAV-MEC to execute quickly, the latency decreases, and the income of the UAV-MEC server increases since there is more time to execute more offloaded data, which increases the system utility. The difference in system utility is noticeable, as the cycles per bit have a direct impact on latency. The system utility ultimately stabilizes at a fixed value after dozens of episodes. The reason for this is that the UAV-MEC consumes more processing resources due to the higher CPU cycles requirements, which would vary with different applications.

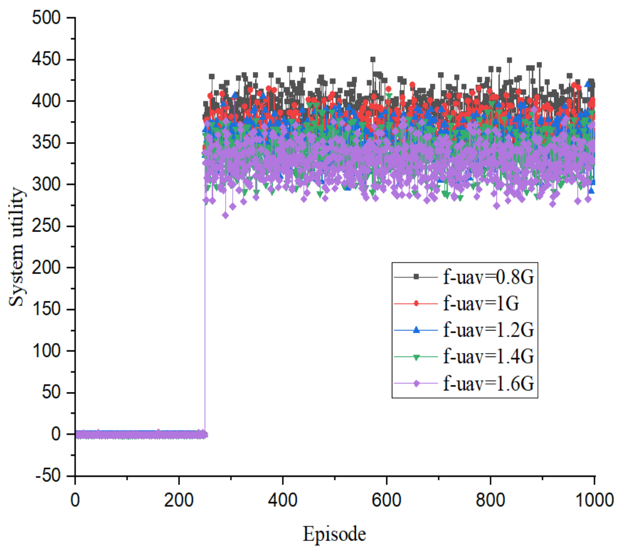

Fig. 6. System utility with different UAV frequencies

As demonstrated in Fig. 6, the system utility increases when the UAV-MEC CPU frequency decreases. The reason is that the UAV-MEC server has more capabilities to process more vehicle data easily to generate more revenue, but using TDMA prevents serving more than one vehicle in each time slot, resulting in more delay in waiting for local execution. After dozens of episodes, the system utility eventually stabilizes at a certain value. On the other hand, this improvement by increasing the server computing capability becomes small in the higher frequency range, where the bottleneck for reducing the execution delay is the transmission delays between vehicles and the UAV-MEC server. As it is illustrated in Fig. 6, the effect of CPU frequency on energy consumption can be converted into the effect of offloading ratio and transmission power, which consequently reduces system utility.

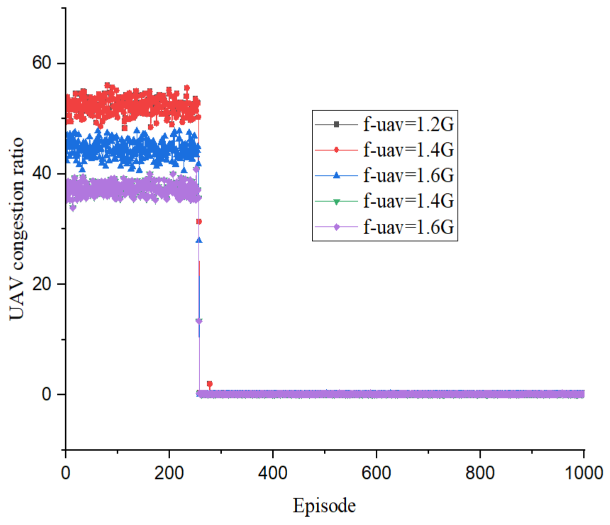

Fig. 7. Congestion ratio with different UAV CPU frequencies

The pricing factor can be used to modify the congestion ratio in Equation (14). Because the relationship between congestion and price is reversed, the price controls the amount of data offloaded to alleviate congestion on the UAV-MEC server. The congestion on the UAV-MEC server increases as the cycles per bit required to analyze one bit of vehicle data increase. As seen in Fig. 7, the CPU capacity of the UAV-MEC has an impact on server congestion because greater resources allow more data to be processed, thereby reducing congestion. Due to the differentiation of vehicle spending dynamics and UAV-MEC processing capability, the latter is motivated to adjust its announced prices to better adapt the volume of offloaded data.

Due to the energy usage and hardware constraints of the UAV-MEC server, the maximum CPU frequency is limited. Rather than simply increasing the CPU frequency of the current UAV-MEC server, expanding the number of UAV-MEC servers is more efficient and practical.

D. *Performance Comparison with Baseline*

For comparison with baseline algorithms, a DDPG learning module is trained within the same environment with a random baseline and a greedy baseline. For the random method, the action is selected randomly from the set of allowable actions. For the greedy method, the agent always selects the action with the maximum reward from the local optimal actions in the replay buffer.

The delay in the intelligent pricing model DDPG converges toward the global optimum reward better than other methods (random and greedy), as shown in Fig. 8.

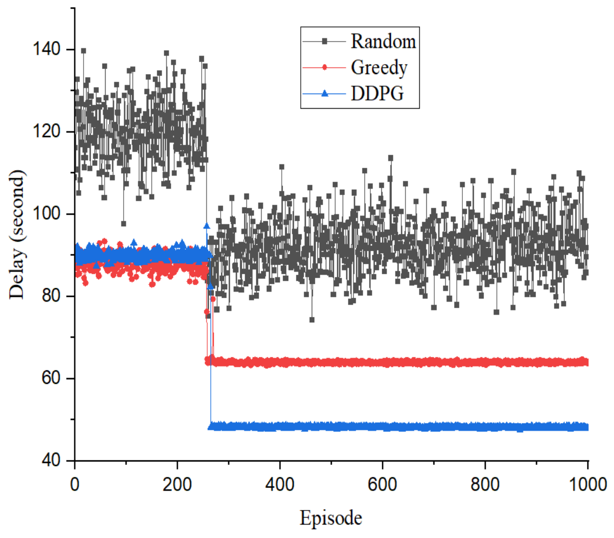

Fig. 8. Comparison of delay with a greedy and random method

The DDPG algorithm produces the shortest possible latency. This finding shows that the DDPG pricing algorithm is capable of learning how to reduce the delay in task offload. The proposed DDPG algorithm outperforms greedy by 26% and random by 51%.

Compared to greedy and random methods, Fig. 9 shows the utility of the system. The proposed technique has lower system utility compared to random due to the constraints in the formulated problem (Equation 15). It requires that the sum of both $\lambda 1$, $\lambda 2$ is less than one, but random does not care. On the other hand, because it relies on local optima, the greedy technique has a lower system utility than the DDPG algorithm.

This is due to the greedy method's insistence on only considering the current task and ignoring the need for future tasks from other vehicles. Normally, the greedy strategy takes all available resources to complete the task, leaving no resources for other vehicles to perform other tasks. The proposed DDPG algorithm outperforms greedy only by 17%.

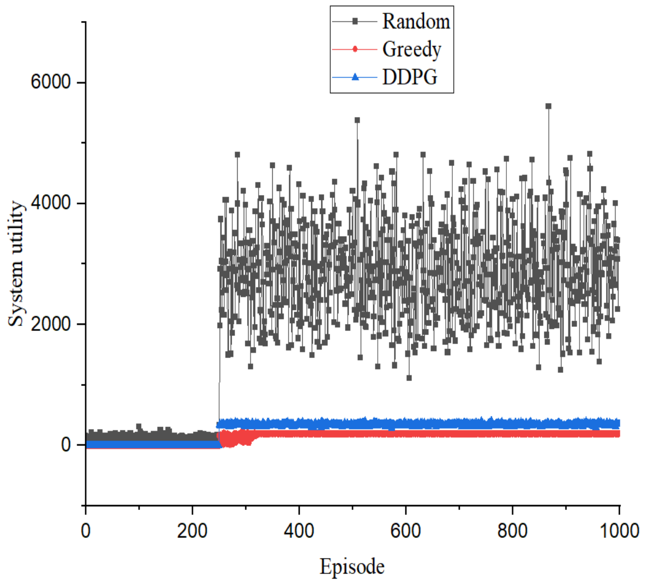

Fig. 9. Comparison of the system utility with a greedy and random method

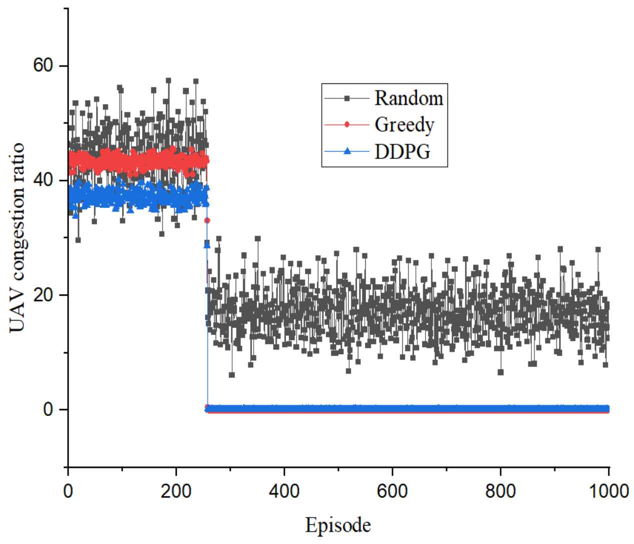

Fig. 10. Comparison of UAV server congestion with greedy and random methods

Fig. 10 indicates that in the DDPG pricing algorithm, UAV server congestion achieves a better reward (less) than greedy and random methods due to the price control. The reward of the random and greedy technique has more congestion than the DDPG pricing algorithm because data offloading to the UAV-MEC server is not balanced. The DDPG algorithm outperforms random by 19%, and in the comparison with greedy, the performance convergence is almost the same.

However, because this paper employs TDMA to schedule vehicle requests, the system utility is reduced when vehicle numbers increase because additional vehicle requests cause congestion, while all vehicle task sizes are 100 Mbps. When 12 or 15 vehicles are used, the entire offloading data will almost completely absorb all UAV resources (1.2 GHz), causing further delays. However, as the number of cycles required to process one bit of vehicle data grows, the system's utility grows. More data may be processed in each time slot, resulting in increased revenue and system utility. The reason for this is that the UAV-MEC server has greater capabilities to process more vehicle data quickly to earn more money, but employing TDMA prevents serving more than one vehicle in each time slot, resulting in more delay for waiting or local execution. In terms of delay and UAV-MEC CPU congestion, the proposed DDPG solution outperforms greedy and random solutions. The DDPG solution is better than greedy in terms of system utility, however random has more system utility because random selection does not follow the restriction in Equation (15).

IV. Concluding Remarks

We studied the use of UAV-mounted MEC to enable data offloading for vehicular networks. In a short period, the UAV-MEC serves numerous vehicles in a hotspot area. In particular, a usage-based payment structure is introduced for the use of the UAV-MEC server's capabilities, and it is properly modeled as an MDP. The proposed dynamic pricing of the offloading model, which is compared to alternative offloading methods, minimizes the UAV's delay and energy burden while increasing the profit of the UAV operator. We also introduced a deep reinforcement learning model (DDPG) for dynamic pricing, which can be used to learn and optimize task offloading in continuous action space. The suggested approach optimizes many criteria, including processing delay, energy usage, and the price disclosed by UAV-MEC. The UAV-MEC server, in particular, seeks to optimize the number of tasks processed within time and energy constraints. With different simulation parameters, the suggested technique achieves better system utility and provides a more reliable offloading strategy by minimizing delay. Compared to the greedy and random methods, the baseline algorithms, our pricing model also gets better performance.

Future research work will focus on the communication link between UAV and vehicles. Due to the fast speed of the vehicle and the UAV, there is no direct mode in downlink and uplink in the multi-cell scenario (multi-hop). As a result, uploading could be done via vehicle-to-vehicle, and downloading could be done via multi-UAV. Another consideration is the multiplexing mode, and this work could be improved by using recent techniques such as OFDMA (Orthogonal Frequency Multiplexing Access) and NOMA (Non-orthogonal Frequency Multiplexing Access). On the other hand, multi-UAV collaboration for the same operator as well as competition between different operators We will also consider the employment of alternative RL algorithms in future studies. In addition, the proposed model will be modified to include additional security aspects in the reputation score of the UAV-MEC server, such as the UAV-MEC server's trust level, security, and privacy-preserving characteristics.

References

[1] Z. Chen, N. Xiao, and D. Han, “A Multilevel Mobile Fog Computing Offloading Model Based on UAV-Assisted and Heterogeneous Network,” *Wirel. Commun. Mob. Comput.*, 2020, doi: 10.1155/2020/8833722.

[2] L. Zhao, K. Yang, Z. Tan, X. Li, S. Sharma, and Z. Liu, “A Novel Cost Optimization Strategy for SDN-Enabled UAV-Assisted Vehicular

Computation Offloading," *IEEE Trans. Intell. Transp. Syst.*, 2020, doi: 10.1109/tits.2020.3024186.

[3] L. Li, M. Siew, T. Q. S. Quek, J. Ren, Z. Chen, and Y. Zhang, "Learning-based priority pricing for job offloading in mobile edge computing," *arXiv*. 2019.

[4] L. Li, M. Siew, T. Q. S. Quek, and Z. Chen, "Optimal Pricing for Job Offloading in the MEC System with Two Priority Classes," 2019, [Online]. Available: http://arxiv.org/abs/1905.07749.

[5] B. Assila, A. Kobbane, and M. El Koutbi, "A Cournot Economic Pricing Model for Caching Resource Management in 5G Wireless Networks," *2018 14th Int. Wirel. Commun. Mob. Comput. Conf.*, pp. 1345–1350, 2018.

[6] L. Zhang *et al.*, "Task Offloading and Trajectory Control for UAV-Assisted Mobile Edge Computing Using Deep Reinforcement Learning," *IEEE Access*, vol. 9, pp. 53708–53719, 2021, doi: 10.1109/ACCESS.2021.3070908.

[7] G. Sun, T. Zhan, B. Gordon Owusu, A. M. Daniel, G. Liu, and W. Jiang, "Revised reinforcement learning based on anchor graph hashing for autonomous cell activation in cloud-RANs," *Futur. Gener. Comput. Syst.*, vol. 104, pp. 60–73, Mar. 2020, doi: 10.1016/j.future.2019.09.044.

[8] J. Zhang, X. Tao, H. Wu, N. Zhang, and X. Zhang, "Deep Reinforcement Learning for Throughput Improvement of Uplink Grant-Free NOMA System," *IEEE Internet Things J.*, 2020, doi: 10.1109/jiot.2020.2972274.

[9] A. Waheed, M. A. Shah, A. Khan, C. Maple, and I. Ullah, "Hybrid task coordination using multi-hop communication in volunteer computing-based vanets," *Sensors*, vol. 21, no. 8, pp. 1–22, 2021, doi: 10.3390/s21082718.

[10] M. Liu and Y. Liu, "Price-Based Distributed Offloading for Mobile-Edge Computing with Computation Capacity Constraints," *IEEE Wirel. Commun. Lett.*, vol. 7, no. 3, pp. 420–423, 2018, doi: 10.1109/LWC.2017.2780128.

[11] M. Nasimi, M. A. Habibi, B. Han, and H. D. Schotten, "Edge-Assisted Congestion Control Mechanism for 5G Network Using Software-Defined Networking," *Proc. Int. Symp. Wirel. Commun. Syst.*, vol. 2018-August, 2018, doi: 10.1109/ISWCS.2018.8491233.

[12] A. Sural *et al.*, "Spatial Resource Allocation in Massive MIMO Communication : From Cellular to Cell-Free," *2018 IEEE Glob. Commun. Conf. GLOBECOM 2018 - Proc.*, vol. 2019, no. 1, pp. 1–6, 2019, doi: 10.1002/adom.201800104.

[13] X. Wang and L. Duan, "Economic Analysis of Unmanned Aerial Vehicle (UAV) Provided Mobile Services," *IEEE Trans. Mob. Comput.*, 2020, doi: 10.1109/tmc.2020.2973088.

[14] G. Mitsis, P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Intelligent dynamic data offloading in a competitive Mobile Edge Computing market," *Futur. Internet*, 2019, doi: 10.3390/fi11050118.

[15] G. Mitsis, E. E. Tsiropoulou, and S. Papavassiliou, "Data offloading in UAV-assisted multi-access edge computing systems: A resource-based pricing and user risk-awareness approach," *Sensors (Switzerland)*, 2020, doi: 10.3390/s20082434.

[16] J. Cheng and D. Guan, "Research on task-offloading decision mechanism in mobile edge computing-based Internet of Vehicle," *Eurasip J. Wirel. Commun. Netw.*, vol. 2021, no. 1, 2021, doi: 10.1186/s13638-021-01984-6.

[17] J. Sun, Q. Gu, T. Zheng, P. Dong, A. Valera, and Y. Qin, "Joint Optimization of Computation Offloading and Task Scheduling in Vehicular Edge Computing Networks," *IEEE Access*, vol. 8, pp. 10466–10477, 2020, doi: 10.1109/ACCESS.2020.2965620.

[18] M. Mehrabi *et al.*, "Mobility- and Energy-Aware Cooperative Edge Offloading for Dependent Computation Tasks," *Network*, vol. 1, no. 2, pp. 191–214, Sep. 2021, doi: 10.3390/network1020012.

[19] S. Li, X. Hu, and Y. Du, "Deep reinforcement learning for computation offloading and resource allocation in unmanned-aerial-vehicle assisted edge computing," *Sensors*, vol. 21, no. 19, 2021, doi: 10.3390/s21196499.

[20] A. Bozorgchenani, S. Maghsudi, D. Tarchi, and E. Hossain, "Computation Offloading in Heterogeneous Vehicular Edge Networks: On-line and Off-policy Bandit Solutions," *IEEE Trans. Mob. Comput.*, pp. 1–16, 2021, doi: 10.1109/TMC.2021.3082927.

[21] N. Xing, Q. Zong, L. Dou, B. Tian, and Q. Wang, "A Game Theoretic Approach for Mobility Prediction Clustering in Unmanned Aerial Vehicle Networks," *IEEE Trans. Veh. Technol.*, 2019, doi: 10.1109/TVT.2019.2936894.

[22] S. Li *et al.*, "Joint Congestion Control and Resource Allocation for Delay-Aware Tasks in Mobile Edge Computing," *Wirel. Commun. Mob. Comput.*, vol. 2021, 2021, doi: 10.1155/2021/8897814.

[23] C. F. Lin *et al.*, "Solar Power Can Substantially Prolong Maximum Achievable Airtime of Quadcopter Drones," *Adv. Sci.*, 2020, doi: 10.1002/advs.202001497.

[24] A. Kulsinskas, P. Durdevic, and D. Ortiz-Arroyo, "Internal Wind Turbine Blade Inspections Using UAVs: Analysis and Design Issues," *Energies*, 2021, doi: 10.3390/en14020294.

[25] Y. Yan, W. Shi, and X. Zhang, "Design of UAV wireless power transmission system based on coupling coil structure optimization," *Eurasip J. Wirel. Commun. Netw.*, 2020, doi: 10.1186/s13638-020-01679-4.

[26] Y. Wang, W. Fang, Y. Ding, and N. Xiong, "Computation offloading optimization for UAV-assisted mobile edge computing: a deep deterministic policy gradient approach," *Wirel. Networks*, vol. 27, no. 4, pp. 2991–3006, May 2021, doi: 10.1007/s11276-021-02632-z.

[27] F. Wu, H. Zhang, J. Wu, and L. Song, "Cellular UAV-to-Device Communications: Trajectory Design and Mode Selection by Multi-Agent Deep Reinforcement Learning," *IEEE Trans. Commun.*, 2020, doi: 10.1109/TCOMM.2020.2986289.

[28] X. Zhang, Y. Zhong, P. Liu, F. Zhou, and Y. Wang, "Resource Allocation for a UAV-Enabled Mobile-Edge Computing System: Computation Efficiency Maximization," *IEEE Access*, 2019, doi: 10.1109/access.2019.2935217.

[29] Z. Zhao, W. Zhou, D. Deng, J. Xia, and L. Fan, "Intelligent Mobile Edge Computing with Pricing in Internet of Things," *IEEE Access*, 2020, doi: 10.1109/ACCESS.2020.2974249.

[30] 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 Things J.*, 2019, doi: 10.1109/JIOT.2018.2878876.

[31] G. Mitsis, E. E. Tsiropoulou, and S. Papavassiliou, "Data offloading in UAV-assisted multi-access edge computing systems: A resource-based pricing and user risk-awareness approach," *Sensors (Switzerland)*, vol. 20, no. 8, Apr. 2020, doi: 10.3390/s20082434.

[32] Q. V. Pham, L. B. Le, S. H. Chung, and W. J. Hwang, "Mobile Edge Computing with Wireless Backhaul: Joint Task Offloading and Resource Allocation," *IEEE Access*, vol. 7, pp. 16444–16459, 2019, doi: 10.1109/ACCESS.2018.2883692.

[33] M. Guo *et al.*, "Hagp: A heuristic algorithm based on greedy policy for task offloading with reliability of mds in mec of the industrial internet," *Sensors*, vol. 21, no. 10, 2021, doi: 10.3390/s21103513.

[34] Y. Guo *et al.*, "Intelligent Offloading Strategy Design for Relaying Mobile Edge Computing Networks," *IEEE Access*, 2020, doi: 10.1109/ACCESS.2020.2972106.

[35] J. H. Anajemba, T. Yue, C. Iwendi, M. Alenezi, and M. Mittal, "Optimal Cooperative Offloading Scheme for Energy Efficient Multi-Access Edge Computation," *IEEE Access*, vol. 8, pp. 53931–53941, 2020, doi: 10.1109/ACCESS.2020.2980196.

[36] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint Computation Offloading and Interference Management in Wireless Cellular Networks with Mobile Edge Computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7432–7445, 2017, doi: 10.1109/TVT.2017.2672701.

[37] M. Mukherjee, V. Kumar, J. Lloret, and Q. Zhang, "Revenue Maximization in Delay-Aware Computation Offloading among Service Providers with Fog Federation," *IEEE Commun. Lett.*, 2020, doi: 10.1109/LCOMM.2020.2992781.

[38] X. Wang, J. Wang, X. Zhang, X. Chen, and P. Zhou, "Joint Task Offloading and Payment Determination for Mobile Edge Computing: A Stable Matching Based Approach," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12148–12161, 2020, doi: 10.1109/TVT.2020.3013622.

[39] L. Li, Y. Li, and R. Li, "Double Auction-Based Two-Level Resource Allocation Mechanism for Computation Offloading in Mobile Blockchain Application," *Mob. Inf. Syst.*, vol. 2021, 2021, doi: 10.1155/2021/8821583.

[40] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint Offloading and Resource Allocation in Vehicular Edge Computing and Networks," 2018, doi: 10.1109/GLOCOM.2018.8648004.

[41] Z. Chen, N. Xiao, and D. Han, "Multilevel task offloading and resource optimization of edge computing networks considering UAV relay and green energy," *Appl. Sci.*, 2020, doi: 10.3390/app10072592.

[42] G. Tefera, K. She, M. Chen, and A. Ahmed, "Congestion-aware adaptive decentralised computation offloading and caching for multiaccess edge computing networks," *IET Commun.*, 2020, doi: 10.1049/iet-com.2020.0630.

[43] S. Chen, L. Li, Z. Chen, and S. Li, "Dynamic Pricing for Smart Mobile Edge Computing: A Reinforcement Learning Approach," *IEEE Wirel. Commun. Lett.*, 2020, doi: 10.1109/LWC.2020.3039863.

[44] B. Cho and Y. Xiao, "Learning-based decentralized offloading decision making in an adversarial environment," *IEEE Trans. Veh. Technol.*, pp. 1–14, 2021, doi: DOI: 10.1109/tvt.2021.3115899.

[45] V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015, doi: 10.1038/nature14236.

[46] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” *31st Int. Conf. Mach. Learn. ICML 2014*, vol. 1, pp. 605–619, 2014.

[47] C. Qiu, Y. Hu, Y. Chen, and B. Zeng, “Deep Deterministic Policy Gradient (DDPG)-Based Energy Harvesting Wireless Communications,” *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8577–8588, 2019, doi: 10.1109/JIOT.2019.2921159.

[48] F. De Vita, D. Bruneo, A. Puliafito, G. Nardini, A. Virdis, and G. Stea, “A deep reinforcement learning approach for data migration in multi-access edge computing,” *10th ITU Acad. Conf. Kaleidosc. Mach. Learn. a 5G Futur. ITU K 2018*, pp. 1–8, 2018, doi: 10.23919/ITU-WT.2018.8597889.

Asrar Ahmed Baktayan received her BSc degree in electronics and telecommunication from Aden University, Yemen, in 2006. She started working for Yemen Mobile-Telecommunication Operator in 2008 as a telecommunication traffic engineer in the core network department and has been working there until now. She is a graduate student of the Master of Information Technology (Networking and Distributed Systems) at Sana'a University. Her research interests are in wireless communication, mobility management, MEC, network slicing, and UAV in 5G networks.

Ibrahim Ahmed Al-Baltah is an Associate Professor in the Department of Information Technology at Sana’a University, where he has been a faculty member since 2015. He is currently the Head of the Information Technology department. He received his BSc in Statistics and Computer Science (2007) from the University of Gezira, Sudan, his MSc in Software Engineering (2009), and his Ph.D. in Software Engineering (2014) from the University of Putra Malaysia. He has published several papers in highly reputable journals. He is a reviewer in some reputed international journals. His research interests are in green software engineering, resilience software engineering, cognitive software engineering, the semantic web, the semantic web of things, and semantic data fusion.

Abdul Azim Abd Ghani received the B.Sc. degree in mathematics/computer science from Indiana State University, the M.Sc. degree in computer science from the University of Miami, and the Ph.D. degree in software engineering from the University of Strathclyde. He is currently a Professor at the Department of Software Engineering and Information System, Universiti Putra Malaysia. His research interests include software measurements, software testing, and software quality.