Optimizing Computation Offloading in 6G Multi-Access Edge Computing Using Deep Reinforcement Learning

Original Scientific Paper

Mamoon M. Saeed
Department of Communications and Electronics Engineering,
University of Modern Sciences (UMS), Sana'a, Yemen
mamoon530@gmail.com

Rashid A. Saeed*
College of Business and Commerce,
Lusail University, Lusail, Qatar
rabdelhaleem@lu.edu.qa

Hashim Elshafie
Department of Computer Engineering,
College of Computer Science,
King Khalid University,
Main Campus, Al Farah,
Abha 61421,
Kingdom of Saudi Arabia, KSA
helshafie@kku.edu.sa

*Corresponding author

Ala Eldin Awouda
Mechanical Engineering Department,
College of Engineering,
Bisha University, Bisha, KSA
aadam@ub.edu.sa

School of Electronics of Engineering,
Faculty of Engineering,
Sudan University of Science and Technology,
Khartoum, Sudan

Zeinab E. Ahmed
Department of Computer Engineering,
University of Gezira,
Wad-Madani, Sudan
Zeinab.e.ahmed@gmail.com

Mayada A. Ahmed
School of Electronics of Engineering,
Faculty of Engineering,
Sudan University of Science and Technology,
Khartoum, Sudan
mayadanott13@gmail.com

Rania A Mokhtar
School of Electronics of Engineering,
Faculty of Engineering,
Sudan University of Science and Technology,
Khartoum, Sudan
ragiliter@gmail.com

Abstract *– One of the most important technologies for future mobile networks is multi-access edge computing (MEC). Computational duties can be redirected to edge servers rather than distant cloud servers by placing edge computing facilities at the edge of the wireless access network. This will meet the needs of 6G applications that demand high reliability and low latency. At the same time, as wireless network technology develops, a variety of computationally demanding and time-sensitive 6G applications appear. These jobs require lower latency and higher processing priority than traditional internet operations. This study presents a 6G multi-access edge computing network design to reduce total system costs, creating a collective optimization challenge. To tackle this problem, Joint Computation Offloading and Task Migration Optimization (JCOTM), an approach based on deep reinforcement learning, is presented. This algorithm takes into consideration several factors, such as the allocation of system computing resources, network communication capacity, and the simultaneous execution of many calculation jobs. A Markov Decision Process is used to simulate the mixed integer nonlinear programming problem. The effectiveness of the suggested algorithm in reducing equipment energy consumption and task processing delays is demonstrated by experimental findings. Compared to other computing offloading techniques, it maximizes resource allocation and computing offloading methodologies, improving system resource consumption. The presented findings are based on a set of simulations done in TensorFlow and Python 3.7 for the Joint Computation Offloading and Task Management (JCOTM) method. Changing key parameters lets us find out that the JCOTM algorithm does converge, with rewards providing a measure of its success compared to various task offloading methods. 15 users and 4 RSUs are placed in the MEC network which faces resource shortages and is aware of users. According to the tests, JCOTM offers a lower average system offloading cost than local, edge, cloud, random computing and a game-theory-based technique. When there are more users and data, JCOTM continues to manage resources effectively and shows excellent speed in processing demands. It can be seen from these results that JCOTM makes it possible to offload efficiently as both server loads and user needs change in MEC environments.*

Keywords: *deep reinforcement learning, sixth generation (6 G), multi-edge computing (MEC), offloading, deep Q-network*

Received: May 8, 2025; Received in revised form: June 7, 2025; Accepted: June 12, 2025

1. INTRODUCTION

The upcoming launch of 6G networks promises a paradigm leap in connectivity in the quickly changing telecoms industry, bringing in a new era of blazingly fast speeds, responsiveness, and dependability [1]. To satisfy the demanding specifications of next-generation networks, it is essential to integrate cutting-edge technologies as the need for high-performance mobile apps keeps growing. Multi-access Edge Computing (MEC) stands out among these technologies as a crucial remedy because it makes it possible to distribute computational jobs closer to the edge of wireless access networks, which lowers latency and improves system efficiency overall [2].

Multi-Access Edge Computing (MEC) is emerging as a transformative technology that significantly enhances network performance by reducing latency through localized data processing. This capability is essential for real-time applications, such as the Internet of Things (IoT) and augmented reality, where rapid response times are crucial. MEC further optimizes bandwidth efficiency by offloading processing tasks from the core network, leading to better resource utilization. The technology also improves user experiences by facilitating seamless interactions and supports a broad spectrum of IoT applications through real-time analytics at the edge. Additionally, MEC enhances security and privacy by minimizing data transmission over networks, thus aiding compliance with privacy regulations. Its scalable architecture accommodates the growing number of devices and applications in today's fast-paced technological environment. Overall, MEC stands out as a pivotal solution in modern networking, optimizing system performance and alleviating pressure on central data centers [3].

Deep Reinforcement Learning (DRL) is a state-of-the-art method for optimizing computation offloading strategies in 6G environments in MEC. Network operators and service providers can intelligently and responsively distribute computing jobs to edge servers by utilizing DRL algorithms' adaptive and self-learning properties [4]. The main requirements of 6G networks, which place a premium on low latency, high dependability, and effective resource use to serve a wide range of cutting-edge applications from augmented reality to driverless cars, are completely met by this integration. In light of this, conducting research and building a deep reinforcement learning-based computation offloading framework designed especially for 6G multi-access edge computing networks is crucial [5].

This framework explores the complex interactions between DRL algorithms and computation offloading techniques to optimize task allocation, improve system performance, and simplify resource management in the context of sophisticated mobile networks [6]. This study aims to push the limits of innovation in mobile communications by investigating the synergies between DRL and computation offloading in the context of 6G MEC networks. It provides a glimpse into the revolutionary potential of AI-driven solutions in influencing the future of network architecture and service delivery [7].

The upcoming 6G technology revolution will reshape different business sectors by improving network connectivity and latency performance alongside the capability to implement time-sensitive software applications. The fundamental development behind network edge transformation rests upon Multi-Access Edge Computing (MEC) for handling computational resources local to the network boundary. Strategic computational load distribution from resource-limited devices to edge computing servers constitutes offloading, so applications and performance gain better efficiency and results [8].

6G networks require effective resource management because of the large number of IoT devices and complex application systems that operate within these networks. Smart devices such as smartphones, along with sensors and autonomous vehicles, use the capability of offloading to transfer complex processing duties to edge servers situated nearby. The device offloading approach helps both devices conserve power and reduce battery drain while speeding up responses and enhancing the user experience altogether [9].

The main parts of offloading execution within MEC consist of many essential elements. The process demands effective decision systems for finding suitable offloading targets among tasks alongside optimal edge server destinations. The implementation of MEC offloading requires an evaluation of the edge server workload together with network performance and the exact needs for each task, including latency tolerance and data magnitude.

The combination of artificial intelligence (AI) with machine learning (ML) methods greatly improves the capability to make offloading decisions. Records from both history and current network situations supply AI algorithms with data to find optimal offloading techniques that enhance the effectiveness of job distribution along with resource organization. The intelligent system delivers both enhanced performance and better 6G network resilience because it rapidly adjusts to both conditions and potential failures [10].

Security, together with privacy issues, represents the highest concern throughout the offloading process. Edge servers require both strong encryption and protected communication protocols to ensure the security of sensitive data transferred from users. 6G networks must guarantee data confidentiality and integrity because such measures are vital for meeting user trust requirements and following regulatory standards when they support a diverse set of applications and multiple devices.

The deployment of offloading systems within 6G

MEC encounters multiple difficulties. Device capability management, alongside network component interoperability and quality of service delivery requirements, make up the implementation challenges of these networks. Offloading methods need to establish coherent processing speed, energy utilization, and networking reliability ratios to deliver continuous user experiences.

The wireless network progression from 5G to 6G technology establishes a new standard of connectivity through heightened speed, together with inferior latency and better capacity. Multi-Access Edge Computing (MEC) serves as the essential tool for network evolution since it distributes computation and storage capabilities near final user locations. An efficient mechanism for complex processing called offloading works effectively because of network decentralization. The transfer of computational operations from smartphone platforms and IoT sensors to enhanced edge servers through offloading describes this process. Application performance can be both optimized and real-time processing and low-latency requirements fulfilled through this essential resource optimization procedure [11].

6G networks must manage unimaginable numbers of linked devices as well as applications, which include autonomous systems and AR and VR applications, because they require increased computational power. Extensive application processing needs exceed the capabilities of local devices, thus making these applications require edge server support. The shifted task execution through offloading activates the edge servers to perform computations with better efficiency and thus improves system performance at large. Several important advantages emerge from offloading, according to the diagram given in Fig. 1.

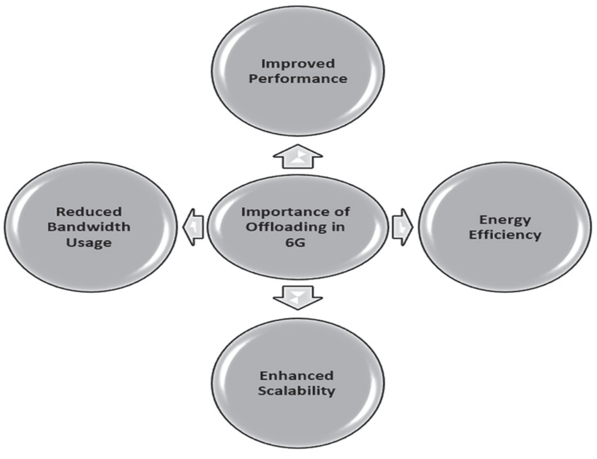

Fig. 1. Offloading Significant Benefits

Edge servers enable applications to lower latency while increasing response times because of their processing capabilities, which serve user satisfaction primarily in real-time applications. Devices with limited resources can save power through edge-based transferring of complex processing demands. The operation of IoT devices that depend on battery power specifically requires this approach. Network scalability becomes possible through offloading because it enables distributed workloads across several edge servers instead of overloading devices and cloud-based resources independently. The offloading process lowers the amount of data that needs to flow to cloud servers for analysis, resulting in reduced bandwidth usage and network congestion occurrences.

1.1. DECISION-MAKING IN OFFLOADING

Making a task of offloading a decision involves evaluating multiple conditions, which should include [12], [13], and [14]:

- Task characteristics and specific parameters such as computational difficulty, together with data volume and response time demands, play an essential role in deciding whether a computation should be transmitted off-device.
- Professional offloading decisions require immediate evaluation of network conditions, including server load, bandwidth availability, and network latency data.
- For successful offloading purposes, it is fundamental to recognize the processing abilities and energy use status of initiating devices.

Advanced algorithms together with models serve as tools to assist in this type of decision-making framework. These may include:

- Advanced offloading algorithms make real-time compensations to fluctuating networks and system work demands for better offloading results.
- The predictive analysis uses AI and ML technologies to develop offloading strategies by processing historical and present network data. The technologies apply past data learning to optimize their functions in distributing resources across teams and assigning tasks more effectively.

1.2. SECURITY AND PRIVACY CONCERNS

Data security and privacy emerge as essential factors after its transport to edge servers. Key considerations include [8]:

- All transmission of sensitive information to edge servers require encryption to stop unauthorized users from accessing the data.
- A secure transmission protocol system must be established to maintain safe data exchange between devices and edge servers.
- The protection of sensitive data requires implementing measures that allow authorized users and devices to access it.
- The protection of user privacy requires organizations to maintain strict obedience to data protection laws like GDPR and HIPAA.

1.3. CHALLENGES IN IMPLEMENTING OFFLOADING

Several key obstacles exist when implementing offloading via MEC in 6G networks [15]:

- The compatibility between different devices and applications, and network components becomes complex because standard protocols and interfaces are needed.
- Developing a favorable quality of service (QoS) remains vital to achieving user satisfaction because sensitive applications require stable bandwidth along with minimal latency.
- Efficient operations of edge server's dependent on resource management create significant problems because of load balancing difficulties and resource allocation demands.
- Modern computational frameworks are needed to execute effective operations for real-time decision-making between abrupt condition changes.

1.4. FUTURE DIRECTIONS

6G technology development indicates that the following directions for MEC offloading will emerge [16]:

- The application of edge intelligence combined with artificial intelligence at horizontal distribution points results in improved decision-making ability that enables dynamic on-the-fly offloading adjustments based on present conditions.
- Federated learning provides decentralized model training that keeps sensitive data on user devices and enables collective learning through decentralized training.
- The adoption of decentralized architectural design brings better resistance and decreases dependency on centralized cloud infrastructure, which enables better offloading outcomes.
- Edge computing operations demand specialized security frameworks that need development according to specific edge needs, since platform evolution will be mandatory.

6G Multi-Access Edge Computing depends on offloading as its core resource optimization and application performance enhancement mechanism [17]. The method of moving computations to edge servers as a strategic step helps handle next-gen application requirements and delivers better energy efficiency and adaptable system capacity [18]. The complete realization of 6G MEC requires attention towards smart calculation management techniques as well as secure protection frameworks and resolution of operational obstacles alongside technological evolution [19].

Through carefully assessing these technologies and their consequences for the mobile ecosystem, our research strives to pave the way for more efficient, intelligent, and responsive network infrastructures capable of addressing the rising needs of the digital age. Here's a summary of the primary contributions:

- The communication and task computation flow are simulated to determine the system delay and energy consumption formula.
- The mixed integer nonlinear programming problem is challenging to solve directly because it is NP-hard. Thus, we convert it into a Markov Decision Process and propose a combined computation offloading and task migration optimization (JCOTM) technique based on deep reinforcement learning.

The JCOTM algorithm's convergence and efficacy are demonstrated by experimental performance. Our suggested approach can lower processing latency and equipment energy usage in various system contexts compared to alternative computation offloading strategies.

The remaining sections of this paper are arranged as follows: In Section III, we outline the joint optimization issue and the 5 G-based 6G user-aware multi-access edge computing network architecture. Section IV introduces the Deep Q-Network and the JCOTM algorithm's comprehensive process. Section V presents the simulation parameters and outcomes, while Section VI wraps up our investigation.

2. RELATED WORK

This part of the study examines previous studies that aimed to improve how computation is distributed in Multi-Access Edge Computing (MEC). The literature is typically organized into binary offloading and making decisions about partial execution.

The tasks can be processed where they are created or sent to the MEC server for completion. [20] analyzes what the best single-user performance is when binary offloading is used in ultra-dense networks. It highlights situations when binary offloading might be useful, and [21] develops an approach using both games and optimization for better results. They help to see the role of server-based processing and when it is more beneficial than running tasks locally, showing the need to decide wisely.

Several experts have used advanced techniques such as reinforcement learning (RL) to manage the complicated issues in MEC. For example, [22] optimizes the use of resources and UAV routes at once, which demonstrates how RL helps save power in fast-changing situations. [23] found that RL can handle some of the MEC's important challenges, such as those related to mobility and managing changing channels.

This framework (MELO, presented in [24]) demonstrates a decision-making system that uses reinforcement learning and formulates the tasks as a Markov Decision Process. It points to more use of machine learning to assist in making choices in the context of

MEC. Alternatively, users with partial offloading can pass some of their work to the MEC server when required. The research in [25] deals with offloading cloud tasks to more than one device, with wireless interference and separable semi-definite relaxation in mind. This technique points out how partial offloading is flexible and able to ensure resources are used well, as different users require them.

Also, techniques such as convex optimization and segmentation optimization are used to optimize resource usage in multi-user MEC systems [26, 27]. They reveal how much effort is put into both minimizing expenses and cutting back on delays that put efficiency and results in balance.

Different approaches, for example, [28], are now considering how load on servers affects energy use, reflecting the increased awareness that workloads and infrastructure affect each other. Unlike the strategies of the papers mentioned in [7], the authors of [29] and [30] stressed that the best way to reduce offloading costs is to pay attention to energy use, processing time, and delay.

[31] and [32] identify that with the advent of 6G, intelligent user edge computing relies heavily on deep reinforcement learning for request offloading and choosing resources. The article [33] also introduced the UMAP algorithm, which further demonstrates the benefits of combining different advanced algorithms to boost MEC performance.

Simply put, while the use of binary offloading helps with straightforward situations, using partial offloading and more advanced techniques allows both the application and network to adapt and respond to what the user needs. The field is seeing how delicate performance, resource management, and what users experience are balanced in MEC.

This work [34] presented the UMAP algorithm that connects handling UAV movement to connecting users with access to a network, all through frequent optimization. With deep reinforcement learning (DRL), the system learns to improve both where UAVs go and how they are associated, which helps reduce the amount of energy used and waiting time in the system. This way of working highlights that DRL is useful in environments that keep adapting, so agents can react to current circumstances.

Even so, due to how complicated DRL models are to train, it can be quite challenging regarding whether they converge and the number of computer resources required, which means they aren’t always practical everywhere. Even though the advancement to closed-form MU transmission power helps efficiency, it may not be suited for different operating settings. To sum up, UMAP reflects important progress in MEC by offloading data, but points out that further study is needed to improve its work in different situations. This stresses the need to blend different optimization strategies to help the entire system perform better.

The proposed system involves 5G technology and 6G user-aware Multi-access Edge Computing network (VAMECN) elements, which consist of 6G users, roadside units, and cloud servers to handle upcoming 5G network offloading functions. The proposed method addresses the reduction of system delays along energy consumption optimization. The proposed solution adopts deep reinforcement learning to create JCOTM for addressing problems through performance demonstrations

3. SYSTEM MODEL AND PROBLEM FORMULATION

As illustrated in Fig. 2, we examine a 5 G-based user-aware Mobile Edge Computing (MEC) network architecture, which comprises N users, M Roadside Units (RSUs), and a cloud server. We define the index sets for users and RSUs as $U = \{1,2,\ldots,N\}$ and $M = \{0,1,2,\ldots, M, M+1$, respectively. Here, $m=0$ represents the local computing device, while $m=M+1$ denotes the cloud server [35]. The indices between 0 and $M+1$ correspond to the edge servers. We assume that the RSUs are uniformly distributed along the road, each covering a consistent area R. Each RSU is equipped with one or more MEC servers, positioning it as an edge computing node.

To effectively simulate the users' trajectories over time, we represent the continuous road as a series of discrete traffic areas. In Fig. 2, a typical urban road network is segmented into PPP discrete areas, indexed by the set $P=\{1, 2,\ldots, P\}$ We will next address the optimization problem related to joint computation offloading and task migration over a defined period T [36]. This period is divided into t_i time slots denote as $T=\{1, 2,\ldots, t_i\}$ at the initial time slot t_0, users are randomly allocated within the network. As users move, they can either remain in their current traffic area or transition to an adjacent one. The transition probability from location l to l' for users can be expressed as $Pr(l' \mid l)$. For instance, $Pr(l \mid l)=0.5$ indicates a 50 % probability that a user will remain in the same location. We assume the probability of moving from l to l' (where $l \neq l'$) is equivalent, allowing us to calculate the position transfer probability as $Pr(l' \mid l) = (1-Pr(l \mid l))/2$ [37]

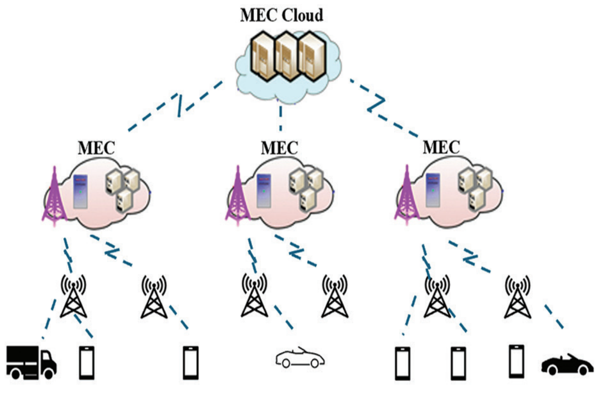

Fig. 2. The architecture of our proposed 6G MEC network

Each user of equipment (UE) is assumed to have a single compute-intensive task that requires processing. There exists a one-to-one correspondence between UEs and users. The *n-th* task can be characterized by a triple α_n, β_n, γ_n, where $n \in v$, α_n denotes the data size of the task, β_n represents the required CPU cycles for task completion, and γ_n indicates the maximum allowable delay. The binary offloading decision is represented by $x_{mn} \in \{0,1\}$, for m $\in M$, $n \in v$. Specifically, x_{mn}=0 indicates that task n will be processed on the local UE, while x_{mn}=1 signifies that the task will be offloaded to the *m-th* MEC server [38].

Notably, when $m=M+1$, the task n is offloaded to the cloud server. The system's offloading decisions at the t^{th} time slot is represented by the set

$X(t)=\{x_{01}(t),\ldots, x_{(M+1)1}(t), x_{02}(t),\ldots, x_{(M+1)N}(t)\}$. It is important to note that each UE can connect to either one RSU or the Base Station (BS) during a time slot [39], thereby necessitating the following constraint:

$$\sum_{m=0}^{M+1} x_{nm}(t) = 1, \forall n \in u \tag{1}$$

Subsequently, we will explore the communication and computation models of the User-Aware MEC Network (UAMECN) system [40], deriving expressions for delay and energy consumption.

3.1. MODEL OF COMMUNICATION

Base station-based communication algorithms create transmission delays that happen when uploading cloud-server data. The rising number of tasks between users causes resource contention that produces network instabilities alongside extended delays. MEC addresses the network bottleneck by establishing server locations that are nearer to user locations [41].

Non-orthogonal multiple Access (NOMA) stands as a vital 5G technology that enables non-orthogonal transmission during signal transmission and incorporates interference data actively while implementing successive interference cancellation (SIC) for accurate signal demodulation at receivers [42]. The receiver implementation of NOMA provides additional complexity compared to OFDMA but delivers higher spectral efficiency. The VAMECN system adopts NOMA for UE-to-BS communication links yet employs OFDMA for UE-to-RSU links because the BS must serve more users [43].

The channel state follows a time-dependent finite continuous value pattern through which the new state appears solely from the previous state. The paper transforms the state values into L discrete levels before representing them as finite-state Markov chains. Channel gain is a crucial parameter for calculating data transmission rates. We denote the channel gain of the wireless link between the user n and RSU m at time t as $\Gamma_n^m(t)$, calculated using the formula [44]:

$$\Gamma_n^m(t) = g_n^m\, d_{n,m}^{-r} \tag{2}$$

Here, g_n^m represents small-scale fading, $d_{(n,m)}$ is the distance between the user n and RSU m, and r is the path loss index. The term $d^{-r}_{n,m}$ signifies path loss. The state space of the Markov chain is represented as $L = \{\Upsilon_1, \Upsilon_2,\ldots, \Upsilon_L\}$, and $\Gamma_n^m(t)$ is classified as Υ_1 when $\Gamma_1^* \le \Gamma_n^m < \Gamma_2^*$; Γ_n^m is quantified as Υ_2 when $\Gamma_2^* \le \Gamma_n^m < \Gamma_3^*$; and so on, Γ_n^m is quantified as Υ_L when $\Gamma_n^m \ge \Gamma_L^*$. ψ_{gs}, $h_s(t)$ is the transition probability that the channel gain shifts from the state g_s to state h_s. Consequently, the following is the $L \times L$ channel state transition probability matrix.

$$\Psi_n^m(t) = [\psi_{gs}, h_s(t)]L \times L \tag{3}$$

Where ψ_{gs}, $h_s(t) = Pr(\Gamma_n^m(t+1) = hs \mid \Gamma_n^m(t) = g_s)$, and $g_s, h_s \in L$. Thus, according to the Shannon formula, the data transmission rate between the user and RSU at time slot t is calculated as follows.

$$\mathrm{u}_n^m(t) = \mathrm{b}_n^m(t)\, log_2(1 + \frac{p_n^m(t)\Gamma_n^m(t)}{\sigma^2}) \tag{4}$$

Where $b_n^m(t)$ the orthogonally allotted bandwidth from RSU m to user n, $m \in M$ and $n \in$ U. $b_n^m(t)$, is denoted by the Gaussian white noise power is represented by σ^2, while transmission power is indicated by $b_n^m(t)$ [45].

Next, we talk about how users and BS communicate. For instance, in the uplink, each UE will be assigned a distinct transmission strength, and signals will be superimposed to send when multiple users are connected to the BS at the same time.

$$X_n(t) = \sqrt{P_n}x_n(t) + \sum_{i \in V, i \neq n} \sqrt{P_i}x_i(t) \tag{5}$$

calculates the superimposed signal, where x_n and x_i stand for the target user n's and other users' transmission signals, respectively. The signal that was received is

$$y_n(t) = \Gamma_n^{M+1}(t)X_n(t) + \sigma^2 \tag{6}$$

After obtaining the data, the BS user rises out of SIC decoding in the decreasing order of channel gain. The interference signal for user n is the sum of the signals with lower equivalent channel gain [46]. In the declining sequence of their channel gains, we assume that N users share the same channel: $\Upsilon_1^{M+1} \ge \Upsilon_2^{M+1} \ge \Upsilon_N^{M+1}$. The data transmission rate $u_n^{M+1}(t)$ and the interference signal $I_n(t)$ If the user is therefore

$$I_n(t) = \sum_{i=n+1}^{N} P_i(\Gamma_n^{M+1}(t))^2 \tag{7}$$

$$u_n^{M+1}(t) = b_n^{M+1}(t)log_2(1 + \frac{P_n^{M+1}(t)(\Gamma_n^{M+1}(t))^2}{\sigma^2 + I_n(t)}) \tag{8}$$

Equation (9) can therefore be used to evenly express the user $n's$ data transmission rate [47].

$$R_n^m(t) = x_{nm}(t)u_n^m(t), \forall n \in U, m \in M \tag{9}$$

The following displays the task $n's$ energy usage and communication delay.

$$T_{nm}^{comm}(t) = \frac{\alpha_n(t)}{R_n^m(t)} \tag{10}$$

$$E_{nm}^{comm}(t) = P_n^m(t)T_{nm}^{comm}(t) \tag{11}$$

$$T_{nm}^{comm}(t) = \sum_{m=1}^{M+1} T_{nm}^{comm}(t)\, x_{nm}(t), \forall n \in U \tag{12}$$

$$E_{nm}^{comm}(t) = \sum_{m=1}^{M+1} E_{nm}^{comm}(t)\, x_{nm}(t), \forall n \in U \tag{13}$$

where α_n (t) is the amount of data left over from the n task. Since $m = 0$ indicates that the work will be processed locally, there is no transmission delay, and no energy consumption, hence in this case, the value of m starts at 1 instead of 0.

3.2. MODEL OF COMPUTATION

User n's task will be sent from the cloud server to the MEC server for computation when x_{nm} (t)=1, $m \in M\backslash\{0\}$. The calculation capability of the server m, commonly referred to as the CPU rate, is represented by the symbol f_m [48]. In particular, the local CPU rate is shown by f_0, and the cloud server's CPU rate is indicated by f_{M+1}. Because edge servers have distinct hardware configurations $f_0 \ll f_m \ll f_{M+1}$, $m \in M\{0, M+1\}$, They are generally more powerful than UE. The distribution of computer resources is not average.

Only one task or one task slice may be completed by each CPU (single core) in each time slot. To simplify the computation model, we assume that every UE has equal entitlement to obtain computing resources [49]. This implies that if n users decide to offload jobs to the same server, the computing resources allotted to each task are $f_{m/n}$. As a result, we can determine the CPU rate assigned to the user n by using the formula

$$f_m^{avg} = \begin{cases} f_0, m = 0 \\ \frac{f_m}{\sum_{i \in u} x_{mi}}, m \in M\backslash\{0\} \end{cases} \tag{14}$$

It is therefore possible to express the processing time for the user n as

$$T_{nm}^{comp}(t) = \sum_{m=0}^{M+1} \frac{\beta_n(t)}{f_m^{avg}} x_{nm}(t), \forall n \ \in U \tag{15}$$

where β_n (t) is the remaining number of CPU cycles needed by the user n during the time slot t. It goes without saying that as server processing capacity rises, computation delay falls. In the meantime, as it influences the CPU time allotted to each user, the server load is also a crucial consideration. The energy used by local equipment in the absence of ask offloading is denoted by E_{nm}^{comp} (t) [50].

$$E_{nm}^{comp}(t) = \mu \beta_n(t)(f_0)^2 T_{nm}^{comp}(t) x_{n0}(t), \forall n \in U \tag{16}$$

Therefore, the energy consumption for user n. Where the effective switched capacitance is represented by μ=10^{-11} [51].

3.3. FORMULATION OF THE PROBLEM

Through the explanation above, we have represented the computation and communication process. Based on our earlier work, we formulate the job completion delay and UE's energy usage as follows.

$$Cost(t) = \xi_t \sum_{n \in U}(T_{nm}^{comm}(t) + T_{nm}^{comp}(t)) + \xi_e \sum_{n \in U}(E_{nm}^{comm}(t) + E_{nm}^{comp}(t)) \tag{17}$$

where ξ_t, $\xi_e \in [0,1]$ are two scalar weights of energy consumption and latency, respectively. Keep in mind that the system latency is the highest of all task computation and communication delays. Consequently, the following is an expression for the joint computation offloading and task migration optimization problem

$$(JCOTM): min \sum_T Cost(t) \tag{18}$$

Subject to:

$$x_{nm} \in \{0,1\}, \forall n \ \in U, m \in M \tag{19}$$

$$\sum_{m=0}^{M+1} x_{nm}(t) = 1, \forall n \ \in U \tag{20}$$

$$\sum_{n \in U, m \in M} b_n^m(t) \ \le B \tag{21}$$

$$\sum_T (T_n^{comm}(t) + T_n^{comp}(t)) \le \gamma_n \forall n \ \in U \tag{22}$$

Table 1 lists the definitions and notations used in this paper. The challenge of optimization, Multiple variable constraints, makes JCOTM a non-convex mixed-integer linear programming issue. The correlation between the variables makes it challenging for us to solve it. As a result, we provide a proposed technique based on Deep Reinforcement Learning (DRL) and model the original problem as a Markov Decision Process (MDP) [52].

Table 1. Notations used in this paper

Notation	Definition
U, N	Index set/number of users
M, m	Index set/number of RSUs
p, P	Index set/number of traffic areas
l, L	The set/number of channel gain states
x_{nm} (t)	x_{nm} $(t) = 1$ if task n is offloaded to server m at time slot t, otherwise, $x_{nm} = 0$
R	Coverage range of one RSU
β_n	Required number of CPU cycles of task n
α_n	Data size of task n
γ_n	max delay limit of task n
γ_l	The l-th state value after the channel gains discretization
$Pr(l' \mid l)$	Transition probability from location l to l' of 6G users
$d_{n,m}^{-r}$	Pass loss
g_n^m	Small-scale fading
Γ_n^m (t)	Channel gain of the communication link between 6G user n and RSU m at time slot t
b_n^m (t)	Bandwidth of the link between 6G user n and RSU m at time slot t
ψ_{gs}, h_s (t)	Transition probability from state h_i to h_j of Γ_n^m (t)
σ^2	Gaussian white noise power
P_n	Transmission power of 6G users n
T_n^{comm}, T_n^{comp}	Communication/computation delay of task n
R_n^m (t)	Data transmission rate from 6G user n to RSU m
f_m	Computation capability of server m
E_n^{comm}, E_n^{comp}	Communication/computation energy consumption of task n
ξ_t, ξ_e	Scalar weight of delay/energy consumption
μ	The effective switched capacitance

4. OPTIMIZING COMPUTATION OFFLOADING BASED DRL

Reinforcement Learning (RL), a subfield of artificial intelligence, is the third machine learning technique, following Unsupervised Learning (UL) and Supervised

Learning (SL). Reinforcement learning involves an agent interacting with its surroundings to learn what actions would result in the greatest reward [53]. In supervised and unsupervised learning, the data is static and does not require interaction with the environment, such as picture recognition. The deep network can learn the difference between samples by iterative training if sufficient samples are provided. However, RL is a dynamic and interactive learning process, and constant contact with the environment also generates the necessary data.

As a result, reinforcement learning incorporates more objects, such as action, environment, state transition probability, and reward function, than supervised learning and unsupervised learning. As a result, when the complexity of a problem approaches that of the actual world, Reinforcement Learning may solve it more effectively [54]. Generally, there are two reinforcement learning algorithms: model-based and model-free. Model in this context refers to the environment's model. The primary distinction between the two algorithms is whether the agent knows the environment model. Model-based has the advantage of allowing the agent to pre-plan the action path based on the features of the known environment. However, it is challenging to get the desired outcome because of the discrepancy between the learned model and the actual world [55].

Consequently, Model-Free is frequently simpler to set up and modify. Value-based, policy-based, and Actor-criticism are the three types of model-free algorithms. Policy-based algorithms model and learn the policy directly, whereas value-based algorithms learn the value function or the action-value function to acquire policy [56]. The benefits of the other two approaches are combined in the Actor-Critic algorithms.

While the critic produces the value of the action, the actor chooses the course of action based on policy. Consequently, the value function and policy impact on one another, accelerating the convergence process. One traditional value-based reinforcement learning algorithm is Q-learning [57]. After learning the Q-values of state-action pairings, the agent chooses the action with the highest Q value. The Q-value, which is the expected reward received by acting a($a \in A$) under state s($s \in S$) at some time, is expressed as $Q^*(s, a) = max_{\pi}$ $E[r_t + \gamma r_{t+1} + \gamma r_{t+2}{}^2 1 + ... | s_t = s, a_t = a, \pi]$ [43]. Q-learning optimizes the policy by updating the complete Q-table in each iteration, using the Q-table to hold the Q-values of all state-action pairs. The formula

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \tag{23}$$

The current state, s, the action was taken at s, s', the state that follows action a, and a', the next possible action at the state s', are all represented in Equation (23). The parameters indicate the learning rate and discount factor α and γ, respectively. The reward results from selecting an action and is denoted by r. Q-learning updates the current Q-value using the maximum Q-value of the subsequent stage. Here, the goal Q-value is denoted by $r + \gamma$ max┬a' $Q(s', a')$, while the estimated Q-value is denoted by $Q(s, a)$ [58]. It goes without saying that when the state and action spaces are too big, the Q-table will grow limitless and require more storage space. A promising approach, DQN (Deep Q-Network), which combines the Q learning algorithm and the deep neural network, addresses the issue.

4.1. DEEP Q-NETWORK ALGORITHM

One significant development in Deep Reinforcement Learning was Google DeepMind Technologies' 2013 proposal of DQN. Figure 3 depicts the DQN structure. DQN has two main advantages over classical Q learning. First, it changes the Q-table updating process into a function-fitting problem, which fits a function rather than a Q-table to produce Q values. In DQN, a deep neural network predicts Q values. Two neural networks predominate.

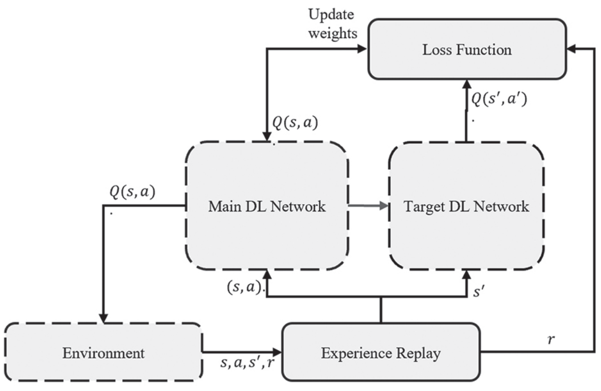

Fig. 3. The DQN structure

One is the main network, which modifies the parameters for every iteration, and the other is the target network, whose parameters are largely fixed [59]. At the same intervals, the target network replicates the parameters from the primary network. As a result, back-propagation only actually trains the primary network. Second, each step of the agent is stored in a unique structure called experience replay, which is denoted by (s, a, r, s'). During each network training cycle, a batch of experiences will be randomly selected from the experience replay for learning. Q-learning can be learned from past and present experiences because it is an off-policy algorithm [60].

Therefore, adding prior experience at random during the learning process will increase the neural network's efficiency and break the correlation between training samples. The following loss function is used for DQN updates at iteration i.

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')}[(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta i)^2] \tag{24}$$

Where the goal Q-value for iteration i is [(r +γ ax┬a')$Q(s', a'; \theta_i^-)$. Until the agent learns to select the best course of action for every state, the neural network is trained, and its parameters are updated by minimizing the value of the loss function in (24), which is the difference between the goal and estimated Q-values [61]. In the following subsection, the specifics of our suggested JCOTM algorithm will be displayed.

4.2. JCOTM ALGORITHM

The optimization issue JCOTM is formulated as a DRL process in this subsection. In this case, the agent is a central management system, which interacts with the surroundings and makes choices. As a result, the agent will broadcast the computation offloading decisions to every UE after gathering status data from servers and automobiles [62]. We must define the three essential components of DQN—the State, Action, and Reward functions—in our algorithm to use it to solve the problem we have been given. Action is the potential behavior of each step, whereas the state is used to represent the environment model. The reward produced by each action, which may be good or negative, is determined using the reward function.

- **State:** $S_n(t)$ represents the condition of the user n at time slot t. The communication state is described by $\Gamma_n^m(t)$, $b_n^m(t)$, while the user state is described by $l_n(t)$, $\alpha_n(t)$, and $\beta_n(t)$. The channel gain and the allotted communication bandwidth between the user n and RSU m at time slot t are denoted by $\Gamma_n^m(t)$ and $b_n^m(t)$, respectively. The traffic area where the user n is at a time slot t is shown by $l_n(t)$,. The remaining data amount is represented by $\alpha_n(t)$, while the necessary number of CPU cycles is represented by $\beta_n(t)$. Consequently, $s_n(t)$ can be written like this:

$$s_n(t) = \{\Gamma_n^1(t), \dots, \Gamma_n^{M+1}(t), \mathrm{b}_n^1(t), \dots, \mathrm{b}_n^{M+1}(t), l_n(t), \alpha_n(t), \beta_n(t)\} \tag{25}$$

- **Action:** Vector $a_n(t) \in R^{M+1}$ indicates whether task n is offloaded to a server m, which is the binary offloading decision. The environment changes from its present state to the next state when the agent selects one action for each time slot t. $a_n(t)$ is defined as follows

$$a_n(t) = \{\mathrm{x}_n^0(t), \mathrm{x}_n^1(t), \dots, \mathrm{x}_n^{M+1}(t)\} \tag{26}$$

- **Reward system:** To determine whether the chosen course of action is good, the environment provides the agent with an indicator value called reward. The optimization objective in this article is to reduce the system cost, which is composed of energy consumption and latency. System cost is hence the reward function.

$$r_n(t) = Cost(t) = \xi_t \sum_{n\in U}(T_{nm}^{comm}(t) + T_{nm}^{comp}(t)) + \xi_e \sum_{n\in U}(E_{nm}^{comm}(t) + E_{nm}^{comp}(t)) \tag{27}$$

Fig. 4 depicts the architecture of the JCOTM algorithm, which is based on deep reinforcement learning. With the same structure, we employ k-deep neural networks (DNN) to forecast binary offloading choices. The action is the neural network's output, while the present state of the environment is its input [63]. We add a decoding layer after the output layer to translate the decimal values into binary. The binary action vector's dimension in our suggested offloading paradigm is $N(M+2)$. We compute the system offloading cost, which is the reward function specified in the preceding material, for each of the output k binary offloading actions. The experience replay unit is initially empty, and a k DNNs start with random parameter values θ_0^k.

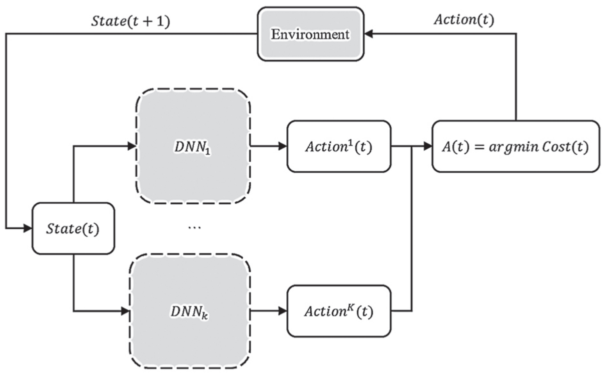

Fig. 4. The architecture of the proposed JCOTM algorithm

The agent chooses the best f-loading action to minimize the reward value in each iteration. The algorithm regularly updates the network parameters and randomly selects a batch of samples from the experience replay unit for training. Gradient descent is used to adjust the parameters to minimize the cross-entropy loss because we switch the DNN's output from predicting the Q-value to action [64].

$$L(\theta_0^k) = -\left[A_t \log\left(f_{\theta_0^k}(S_t)\right) + (1 - A_t) \log\left(1 - f_{\theta_0^k}(S_t)\right)\right] \tag{28}$$

Algorithm 1 displays the JCOTM algorithm's pseudo code.

Algorithm 1. The JCOTM Algorithm is based on DRL.

```
1:  Input: status of the environment $State(t)$
2:  Output: decision for offloading $Action(t)$
3:  Initialization:
4:      initialize environment state $State(t)$
5:      The offloading procedure begins by using
        an identically structured $k$ DNNs.
6:      initialize experience replay.
7:  for $t = 0,1,...,T$: do
8:      Input the current environment state $S_t$.
9:      Get the outputs of each DNN.
10:     Apply decoding techniques to the output
        values to obtain $A_t^i$.
11:     The offloading decision $A_t$ is selected
        through $arg\ min\ R_t$ where by
        $R_t = arg\ min_{i=1,...,k}\ Q(S_t, A_t^i)$.
```

```
12:     After execution of $Action(t)$ environment
        progresses to its new status $S_{t+1}$.
13:     The experience reply receives a tuple
        $A_t, R_t, S_t, S_{t+1}$.
14:     The parameters within DNNs get updated
        through data from randomly chosen
        training batches.
15: end for
```

5. ANALYSIS OF SIMULATION

To assess the effectiveness of our suggested JCOTM method, we create various simulation tests in this part. TensorFlow and Python 3.7 serve as the foundation for the simulation environment. First, by modifying the model's important parameters, we confirm that the JCOTM algorithm is convergent. Next, we assess the development of the deep reinforcement learning-based system offloading technique by comparing the average system offloading cost of JCOTM with other task offloading policies.

To construct a resource-constrained user-aware MEC network, we set the number of users $N = 15$ and the number of RSUs $M = 4$. Each $P = 4$ traffic region that makes up the route has a single RSU with a coverage diameter of $R = 1$ km. Each UE, edge server, and cloud server has CPU frequencies of 0.6×10^9, 1×10^{10}, and 1×10^{12}, respectively [65]. The Gaussian white noise power σ^2 is -88dB, and the overall bandwidth B is 10 MHz. The data size of job n αn is assumed to be between 10M and 30M, and $\rho = 960$ Cycles/Byte is the number of CPU cycles needed for one byte [66]. Table 2 is a list of some important parameters. We will then do our simulation exercises and examine the outcomes.

Table 2. Simulation parameters

Parameter	Value
f_0	0.6 GHz
f_m	10 GHz
f_{M+1}	1 THz
α_n	[10, 30] Mb
σ^2	-88 dB
ρ	960 Cycles/Byte
B	10 MHz
R	1 km
P	4
M	4
N	15

5.1. JCOTM CONVERGENCE

JCOTM convergence has been measured by the reward ratio, by dividing the cost of the optimal offloading policy by enumerating the cost of the policy created by JCOTM, as the assessment indicator to confirm the convergence of JCOTM [67]. Consequently, the algorithm performs better the closer the reward ratio is to 1. It is defined as follows:

$$Reward\ Ratio\ = \frac{C_{policy}}{C_{JCOTM}} \tag{29}$$

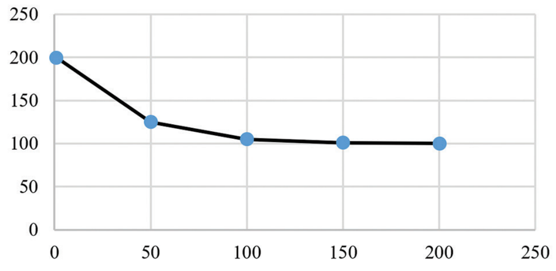

Fig. 5. Convergence of JCOTM Offloading Cost Over Iterations

Fig. 4 shows the cost of JCOTM keeps falling and eventually stabilizes as the number of simulations rises, in an environment with 20 users. As time goes on, JCOTM gets closer to the best policy than before.

5.2. PERFORMANCE OF DIFFERENT OFFLOADING POLICIES

This paragraph evaluates different offloading computing methods within the context. Our suggested method, JCOTM, joins the following different offloading rules, which form the basis of this analysis.

1. UE performs all its tasks individually without server transfers when performing local computing. The system cost results from the weighted sum of energy expended by devices, together with local computational delays.
2. Using edge servers as processing centers is known as edge computing, where all operations are transferred instead of running on the local devices [68]. The system cost includes computational delay and transmission delay, together with UE energy consumption that happens when data needs to be transferred. The concept of Edge computing in this application means all workloads are sent to execute on a single MEC server.
3. Cloud Computing works just like conventional operator cloud services, where all functions get processed on cloud-based servers. The distance between users and the cloud server results in higher transmission delays alongside increased energy consumption.
4. The random computing policy makes offloading decisions by selecting from available options randomly. A single operation can receive processing either within the local network or an edge server, or through the cloud infrastructure.
5. The VAMECN compute offloading problem receives dynamic non-cooperative game model analysis through DGTA, which leads to the determination of Nash Equilibrium solutions [69]. Each user receives a chance to select their optimal offloading strategy

per DGTA algorithm iteration since this method relies on game theory. Figure 6 shows the system offloading expenses of the six different policies while the user count varies. It is evident that when the number of users for all policies increases, the system cost progressively increases as well.

Policies for offloading and the inferior offloading performance are attained by DGTA. Additionally, random computing outperforms edge and local computing, but cloud computing outperforms random computing. Furthermore, the local computing strategy's offloading cost is higher than edge computing's when there are fewer than sixteen users, while the opposite is true when there are more than sixteen. The rationale is that if several workloads are offloaded to the same MEC server [70], there will be less computing power available for each user, which will raise the cost of computation.

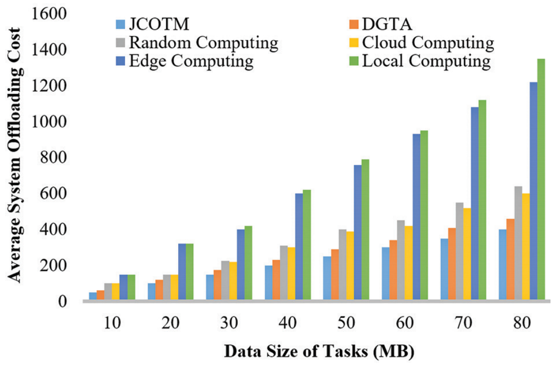

Fig. 6. The average system offloading cost is compared to varying user counts

The average system offloading cost under various job data sizes is compared in Fig. 7. Here, we used Fig. 7 to independently determine the offloading cost. Average system offloading costs for varying user counts are compared. The 10MB to 80MB data size range. The average cost of computing offloading progressively rises as task data sizes increase. JCOTM outperforms the other offloading policies since it optimizes the allocation of system resources [71], whereas other policies either do not accomplish the best allocation of system resources or only use a specific type of computing resources. Local computing has the highest offloading cost, followed by edge computing.

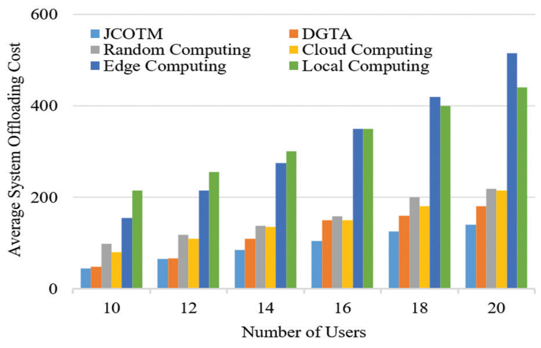

Fig. 7. Average system offloading cost comparison for activities with varying data volumes

On the other hand, cloud computing, random computing, and DGTA have reduced average system offloading costs. The effect of varying numbers of MEC servers on the average system offloading cost is seen in Fig. 8.

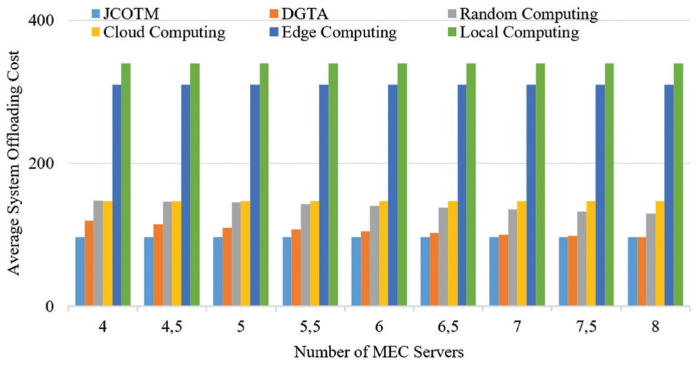

Fig. 8. Comparison of the average cost of system offloading for varying MEC server counts

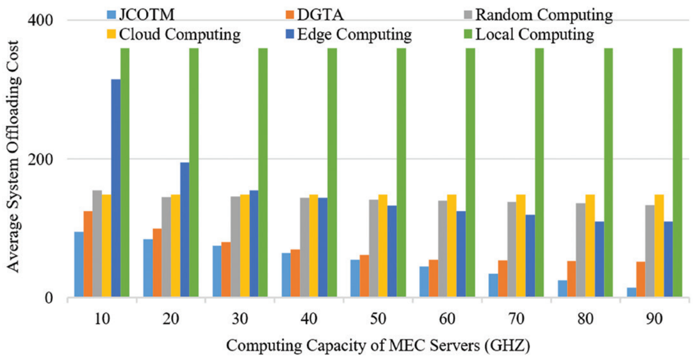

Fig. 9. The average system offloading cost is compared to MEC servers with varying computational capacities

Naturally, the curves are straight lines parallel to the x-axis because local and cloud computing are unaffected. Using Figure 7, as illustrated in Figure 8. Comparison of average system offloading costs for tasks with varying task data sizes [72]. The edge computing offloading cost curve resembles a horizontal straight line as the number of MEC servers increases.

Since all jobs are offloaded to a single MEC server for computation under the edge computing policy, increasing the number of MEC servers has a minimal effect on offloading costs, making this easy to explain [73]. The curves drop as the number of MEC servers grows since additional MEC servers can minimize mode computing delay for random computing and DGTA rules. The chart shows that the average JCOTM system offloading cost is nearly unaffected by the quantity of MEC servers.

One argument is that the cost curve does not exhibit a noticeable downward trend because the resource-constrained environment we have simulated can only satisfy the computational needs of every user.

The average system offloading cost for MEC servers with varying computing capacities is compared in Figure 9. Likewise, the computing power of MEC servers has little bearing on cloud or local computing. The chart indicates that the lower the offloading cost, the greater the computational capability of MEC servers. Additionally, when MEC servers' processing power increases, the offloading cost's rate of decline progressively slows down.

Compared to the other policies, JCOTM has a lower average system offloading cost. Additionally, edge computing outperforms cloud computing in terms of offloading costs when MEC server processing power reaches above 30GHz, and when it reaches over 40GHz, edge computing. We infer that the average system offloading cost is mostly determined by the computational capacity of MEC servers.

6. CONCLUSION

This paper addresses the joint multi-user computation offloading and task migration optimization problem under user-aware Multi-access Edge Computing networks. It considers several factors, including the distribution of system computing resources, communication bandwidth, and concurrent multiple computation tasks. It then suggests a deep reinforcement learning-based JCOTM algorithm to reduce system latency and energy consumption. To increase communication rate and quality and decrease communication latency, we completely consider the Non-Orthogonal Multiple Access technology in the upcoming 5G network during the problem modeling process.

The algorithm abstracts the offloading policy and system resources into the binary action vector and environment state, respectively. Additionally, a deep neural network is used to forecast offloading choices. Until the best offloading choice is found, the agent uses several iterative training courses to perceive the condition of the environment. We create simulation experiments to assess the algorithm's performance and convergence. The simulation findings demonstrate that JCOTM outperforms other offloading strategies under various experiment situations and converges with varying algorithm parameter values. As a result, the technique we suggested can successfully lower the VAMECN system's overall delay and energy usage.

7. REFERENCES

[1] S. Alamuri et al. “Transition From 5G to 6G Communication Technologies: Workforce Evolution and Skill Development Needs”, 5G/6G Advancements in Communication Technologies for Agile Management, IGI Global Scientific Publishing, 2025, pp. 117-142.

[2] S. R. Alkaabi, M. A. Gregory, S. Li, “Multi-access edge computing handover strategies, management, and challenges: a review”, IEEE Access, Vol. 12, 2024, pp. 4660-4673.

[3] A. Elnaim et al. “Energy Consumption for Cognitive Radio Network Enabled Multi-Access Edge Computing”, Proceedings of the 3rd International Conference on Emerging Smart Technologies and Applications, Taiz, Yemen, 10-11 October 2023, pp. 1-5.

[4] P. Wei, K. Guo, J. Wang, W. Feng, S. Jin, “Reinforcement learning-empowered mobile edge computing for 6G edge intelligence”, IEEE Access, Vol. 10, 2022, pp. 65156-65192.

[5] M. M. Saeed et al. “Anomaly detection in 6G networks using machine learning methods”, Electronics, Vol. 12, No. 15, 2023, p. 3300.

[6] M. Khani, M. M. Sadr, S. Jamali, “Deep reinforcement learning-based resource allocation in multi-access edge computing”, Concurrency and Computation: Practice and Experience, Vol. 36, No. 15, 2024, p. e7995.

[7] M. M. Saeed, E. S. Ali, R.A. Saeed, “Data-driven techniques and security issues in wireless networks”, Data-Driven Intelligence in Wireless Networks, CRC Press, 2023, pp. 107-154.

[8] M. I. Khattak, H. Yuan, A. Khan, A. Ahmad, I. Ullah, M. Ahmed, “Evolving Multi-Access Edge Comput-

ing (MEC) for Diverse Ubiquitous Resources Utilization: A Survey", Telecommunication Systems, Vol. 88, No. 2, 2025, pp. 1-41.

[9] M. M. Saeed et al. "Multi-Access Edge Computing Using Intelligent Mobile User Resource Allocation In 6G", Proceedings of the IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering, Tripoli, Libya, 19-21 May 2024.

[10] J. C. Cepeda-Pacheco, M. C. Domingo, "Reinforcement Learning and Multi-Access Edge Computing for 6G-Based Underwater Wireless Networks", IEEE Access, Vol. 13, 2025, pp. 60627-60642.

[11] R. Dulot, L. Mendiboure, Y. Pousset, V. Deniau, F. Launay, "non-orthogonal multiple access for offloading in multi-access edge computing: A survey", IEEE Access, Vol. 11, 2023, pp. 118983-119016.

[12] R. O. Ogundokun et al. "Non-Orthogonal Multiple Access Enabled Mobile Edge Computing in 6G Communications: A Systematic Literature Review", Sustainability, Vol. 15, No. 9, 2023, p. 7315.

[13] S. Jahandar et al. "Handover Decision with Multi-Access Edge Computing in 6G Networks: A Survey", Results in Engineering, 2025, p. 103934.

[14] T. K. Rodrigues, J. Liu, N. Kato, "Offloading decision for mobile multi-access edge computing in a multi-tiered 6G network", IEEE Transactions on Emerging Topics in Computing, Vol. 10, No. 3, 2021, pp. 1414-1427.

[15] H. Hui, Q. Ye, Y. Zhou, "6G-empowered offloading for realtime applications in multi-access edge computing", IEEE Transactions on Network Science and Engineering, Vol. 10, No. 3, 2022, pp. 1311-1325.

[16] L. Zhao et al. "Open-source multi-access edge computing for 6G: Opportunities and challenges", IEEE Access, Vol. 9, 2021, pp. 158426-158439.

[17] M. Hassan, K. Hamid, R. A. Saeed, H. Alhumyani, A. Alenizi, "Reconfigurable Intelligent Surfaces in 6G mMIMO NOMA Networks: A Comprehensive Analysis", International Journal of Electrical and Computer Engineering Systems, Vol. 16, No. 2, 2025, pp. 87-97.

[18] I. Begić, A. S. Kurdija, Ž. Ilić, "A Framework for 5G Network Slicing Optimization using 2-Edge-Connected Subgraphs for Path Protection", International Journal of Electrical and Computer Engineering Systems, Vol. 15, No. 8, 2024, pp. 675-685.

[19] W. Daniar, K. Asmoro, S. Y. Shin, "Joint optimization of phase shift and task offloading for RIS-assisted multi-access edge computing in beyond 6G communication", ICT Express, Vol. 10, No. 3, 2024, pp. 620-625.

[20] X. Chen et al. "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning", IEEE Internet of Things Journal, Vol. 6, No. 3, 2018, pp. 4005-4018.

[21] J. Zhang, W. Xia, F. Yan, L. Shen, "Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing", IEEE Access, Vol. 6, 2018, pp. 19324-19337.

[22] H. Sun, J. Wang, D. Yong, M. Qin, N. Zhang, "Deep reinforcement learning-based computation offloading for mobile edge computing in 6G", IEEE Transactions on Consumer Electronics, Vol. 70, No. 4, 2024, pp. 7482-7493.

[23] H. Huang, Q. Ye, Y. Zhou, "6G-empowered offloading for real-time applications in multi-access edge computing", IEEE Transactions on Network Science and Engineering, Vol. 10, No. 3, 2022, pp. 1311-1325.

[24] M.-H. Chen, B. Liang, M. Dong, "Joint offloading decision and resource allocation for multi-user multi-task mobile cloud", Proceedings of the IEEE International Conference on Communications, Kuala Lumpur, Malaysia, 22-27 May 2016, pp. 1-6.

[25] X. Chen, L. Jiao, W. Li, X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing", IEEE/ACM Transactions on Networking, Vol. 24, No. 5, 2015, pp. 2795-2808.

[26] C. You, K. Huang, H. Chae, B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading", IEEE Transactions on Wireless Communications, Vol. 16, No. 3, 2016, pp. 1397-1411.

[27] J. Ren, G. Yu, Y. Cai, Y. He, "Latency optimization for resource allocation in mobile-edge computation offloading", IEEE Transactions on Wireless Communications, Vol. 17, No. 8, 2018, pp. 5506-5519.

[28] Y. Dai, D. Xu, S. Maharjan, Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing", IEEE Transactions on Vehicular Technology, Vol. 67, No. 12, 2018, pp. 12313-12325.

[29] L. Huang, X. Feng, L. Zhang, L. Qian, Y. Wu, "Multi-server multi-user multi-task computation offloading for mobile edge computing networks", Sensors, Vol. 19, No. 6, 2019, p. 1446.

[30] A. R. Askhedkar, B. Chaudhari, R. A. Saeed, H. Alhumyani, A. Alenizi, "Performance of TVWS-based LoRa Transmissions using Multi-Armed Bandit", International Journal of Electrical and Computer Engineering Systems, Vol. 15, No. 9, 2024, pp. 759-769.

[31] D. Wang, H. Qin, B. Song, X. Du, M. Guizani, "Resource allocation in information-centric wireless networking with D2D-enabled MEC: A deep reinforcement learning approach", IEEE Access, Vol. 7, 2019, pp. 114935-114944.

[32] A. A. Elnaim et al. "Energy Consumption for Cognitive Radio Network Enabled Multi-Access Edge Computing", Proceedings of the 3rd International Conference on Emerging Smart Technologies and Applications, Taiz, Yemen, 10-11 October 2023, pp. 1-5.

[33] Y. Liu, H. Yu, S. Xie, Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in 6G user edge computing and networks", IEEE Transactions on Vehicular Technology, Vol. 68, No. 11, 2019, pp. 11158-11168.

[34] J. Chen et al. "Deep reinforcement learning based resource allocation in multi-UAV-aided MEC networks", IEEE Transactions on Communications, Vol. 71, No. 1, 2022, pp. 296-309.

[35] W. Zhang, Z. Zheng, "Task migration for mobile edge computing using deep reinforcement learning", Future Generation Computer Systems, Vol. 96, 2019, pp. 111-118.

[36] B. Di, L. Song, Y. Li, G. Y. Li, "Non-orthogonal multiple access for high-reliable and low-latency V2X communications in 5G systems", IEEE journal on selected areas in communications, Vol. 35, No. 10, 2017, pp. 2383-2397.

[37] Y. Wu et al. "NOMA-assisted multi-access mobile edge computing: A joint optimization of computation offloading and time allocation", IEEE Transactions on Vehicular Technology, Vol. 67, No. 12, 2018, pp. 12244-12258.

[38] Y. Saito et al. "Non-orthogonal Multiple Access (NOMA) for Cellular Future Radio Access", Proceedings of the IEEE 77th Vehicular Technology Conference, Dresden, Germany, 2-5 June 2013, pp. 1-5.

[39] Z. Ding, P. Fan, H. V. Poor, "Impact of user pairing on 5G nonorthogonal multiple-access downlink transmissions", IEEE Transactions on Vehicular Technology, Vol. 65, No. 8, 2015, pp. 6010-6023.

[40] G. Kivanc, H. Liu, "Computationally efficient bandwidth allocation and power control for OFDMA", IEEE transactions on wireless communications, Vol. 2, No. 6, 2003, pp. 1150-1158.

[41] M. Barakat et al. "Performance Evaluation of Multi-Access Edge Computing for Blended Learning Services", Proceedings of the 21st Learning and Technology Conference, Jeddah, Saudi Arabia, 15-16 January 2024, pp. 197-202.

[42] Z. Wu, D. Yan, "Deep reinforcement learning-based computation offloading for 5G 6G user-aware multi-access edge computing network", China Communications, Vol. 18, No. 11, 2021, pp. 26-41.

[43] V. Mnih et al. "Human-level control through deep reinforcement learning", Nature, Vol. 518, 2015, pp. 529-533.

[44] H. W. Kuhn, "Extensive games and the problem of information", Contributions to the Theory of Games, Vol. 2, No. 28, 1953, pp. 193-216.

[45] M. M. Saeed et al. "Task reverse offloading with deep reinforcement learning in multi-access edge computing", Proceedings of the 9th International Conference on Computer and Communication Engineering, Kuala Lumpur, Malaysia, 15-16 August 2023, pp. 322-327.

[46] M. K. Hasan et al. "An improved binary spider wasp optimization algorithm for intrusion detection for industrial Internet of Things", IEEE Open Journal of the Communications Society, Vol. 6, 2024, pp. 2926-2944.

[47] Z. E. Ahmed et al. "TinyML network applications for smart cities", TinyML for Edge Intelligence in IoT and LPWAN Networks, Elsevier, 2024, pp. 423-451.

[48] M. M. Saeed, R. A. Saeed, Z. E. Ahmed, “TinyML for 5G networks”, TinyML for Edge Intelligence in IoT and LPWAN Networks, Elsevier, 2024, pp. 167-229.

[49] S. Khan et al. “Optimizing deep neural network architectures for renewable energy forecasting”, Discover Sustainability, Vol. 5, No. 1, 2024, p. 394.

[50] Z. Chen, F. Wang, X. Zhang, “Joint Optimization for Cooperative Service-Caching, Computation-Offloading, and Resource-Allocations Over EH/MEC 6G Ultra-Dense Mobile Networks”, IEEE Transactions on Wireless Communications, 2025. (in press)

[51] Z. Hu et al. “DRL-Based Trajectory Optimization and Task Offloading in Hierarchical Aerial MEC”, in IEEE Internet of Things Journal, Vol. 12, No. 3, 2025, pp. 3410-3423.

[52] S. Zhang et al. “Stackelberg Game-Based Multi-Agent Algorithm for Resource Allocation and Task Offloading in MEC-Enabled C-ITS”, IEEE Transactions on Intelligent Transportation Systems, 2025. (in press)

[53] J. Carlos, M. C. Domingo, “Reinforcement Learning and Multi-Access Edge Computing for 6G-Based Underwater Wireless Networks”, IEEE Access, Vol. 13, 2025, pp. 60627-60642.

[54] M. Hevesli, A. M. Seid, A. Erbad, M. Abdallah “Multi-Agent DRL for Queue-Aware Task Offloading in Hierarchical MEC-Enabled Air-Ground Networks”, IEEE Transactions on Cognitive Communications and Networking, 2025. (in press)

[55] J. Bi et al. “Energy-Minimized Partial Computation Offloading in Satellite–Terrestrial Edge Computing Networks”, IEEE Internet of Things Journal, Vol. 12, No. 5, 2025, pp. 5931-5944.

[56] M. Ahmed et al. “Advancements in RIS-Assisted UAV for Empowering Multiaccess Edge Computing: A Survey”, IEEE Internet of Things Journal, Vol. 12, No. 6, 2025, pp. 6325-6346.

[57] D. M. Rani, Supreethi K P, B. B. Jayasingh, “Deep Reinforcement Learning for Dynamic Task Scheduling in Edge-Cloud Environments”, International Journal of Electrical and Computer Engineering Systems, Vol. 15, No. 10, 2024, pp. 837-850.

[58] J. Pacheco, “Contribution to the enhancement of IoT-based application development and optimization of underwater communications, by artificial intelligence, edge computing, and 5G networks and beyond, in smart cities/seas”, Department of Network Engineering, Polytechnic University of Catalonia, Barcelona, Spain, 2024, PhD Thesis.

[59] Z. Wang et al. “AUV-assisted node repair for IoUT relying on multiagent reinforcement learning”, IEEE Internet Things Journal, Vol. 11, No. 3, 2024, pp. 4139-4151.

[60] J. Cao et al. “Multi-agent reinforcement learning charging scheme for underwater rechargeable sensor networks”, IEEE Communication Letters, Vol. 28, No. 3, 2024, pp. 508-512.

[61] Z. Zhao et al. “A transmission-reliable topology control framework based on deep reinforcement learning for UWSNs”, IEEE Internet Things Journal, Vol. 10, No. 15, 2023, pp. 13317-13332.

[62] Z. Zhang et al. “Environment- and energy-aware AUV-assisted data collection for the Internet of Underwater Things”, IEEE Internet Things J., Vol. 11, No. 15, 2024, pp. 26406-26418.

[63] X. Hou et al. “Environment-aware AUV trajectory design and resource management for multi-tier underwater computing”, IEEE Journal on Selected Areas in Communications, Vol. 41, No. 2, 2023, pp. 474-490.

[64] K. G. Omeke, M. Mollel, S. T. Shah, L. Zhang, Q. H. Abbasi, M. A. Imran, “Toward a sustainable Internet of Underwater Things based on AUVs, SWIPT, and reinforcement learning”, IEEE Internet Things Journal, Vol. 11, No. 5, 2024, pp. 7640-7651.

[65] P. Q. Truong et al. “Computation Offloading and Resource Allocation Optimization for Mobile Edge Computing-Aided UAV-RIS Communications”, IEEE Access, Vol. 12, 2024, pp. 107971-107983.

[66] Y. Sadovaya et al. “Enhancing Service Continuity in Non-Terrestrial Networks via Multi-Connectivity Offloading”, IEEE Communications Letters, Vol. 28, No. 10, 2024, pp. 2333-2337.

[67] K. Ali, “Multiradio Parallel Offloading in Multi-access Edge Computing: Optimizing Load Shares, Scheduling, and Capacity”, IEEE Internet of Things Journal, Vol. 11, No. 3, 2024, pp. 4047-4062.

[68] Z. Zhang, Y. Xiao, Z. Ma, M. Xiao, Z. Ding, X. Lei, “6G Wireless Networks: Vision, Requirements, Architec-

ture, and Key Technologies", IEEE Vehicular Technology Magazine, Vol. 14, No. 3, 2019, pp. 28-41.

[69] Z. Wang et al. "AUV-Assisted Node Repair for IoUT Relying on Multiagent Reinforcement Learning", IEEE Internet of Things Journal, Vol. 11, No. 3, 2024, pp. 4139-4151.

[70] J. Cao et al. "Multi-Agent Reinforcement Learning Charging Scheme for Underwater Rechargeable Sensor Networks", IEEE Communications Letters, Vol. 28, No. 3, 2024, pp. 508-512.

[71] P. G. Satheesh, T. Sasikala, "FEDRESOURCE: Federated Learning Based Resource Allocation in Modern Wireless Networks", International Journal of Electrical and Computer Engineering Systems, Vol. 14, No. 9, 2023, pp. 1023-1030.

[72] Z. Zhang et al. "Environment- and Energy-Aware AUV-Assisted Data Collection for the Internet of Underwater Things", IEEE Internet of Things Journal, Vol. 11, No. 15, 2024, pp. 26406-26418.

[73] X. Hou et al. "Environment-Aware AUV Trajectory Design and Resource Management for Multi-Tier Underwater Computing", IEEE Journal on Selected Areas in Communications, Vol. 41, No. 2, 2023, pp. 474-490.