Hindawi
Mobile Information Systems
Volume 2022, Article ID 5051496, 16 pages
https://doi.org/10.1155/2022/5051496

Research Article

Three-Tier Computing Platform Optimization: A Deep Reinforcement Learning Approach

Chidiebere Sunday Chidume,[1] **Solomon Inalegwu Okopi**,[2] **Taiwo Sesay**,[3] **Irene Simon Materu**,[4] **and Theophilus Quachie Asenso**[5]

[1]*Department of Information and Communication Engineering, Xi'an Jiaotong University, Xi'an, China*
[2]*Department of Control Science and Engineering, Xi'an Jiaotong University, Xi'an, China*
[3]*Department of Communication and Transportation Engineering, Chang'an University, Xi'an, China*
[4]*Department of Cyber Security,Xidian University, Xi'an, China*
[5]*Department of Biostatistics, Institute of Basic Medical Sciences University of Oslo, Norway*

Correspondence should be addressed to Chidiebere Sunday Chidume; chidumechidieberesunday@gmail.com

Received 4 September 2021; Accepted 18 May 2022; Published 10 June 2022

Academic Editor: Mario Muñoz-Organero

The increasing number of computing platforms is critical with the increasing trend of delay-sensitive complex applications with enormous power consumption. These computing platforms attach themselves to multiple small base stations and macro base stations to optimize system performance if appropriately allocated. The arrival rate of computing tasks is often stochastic under time-varying wireless channel conditions in the mobile edge computing Internet of things (MEC IoT) network, making it challenging to implement an optimal offloading scheme. The user needs to choose the best computing platforms and base stations to minimize the task completion time and consume less power. In addition, the reliability of our system in terms of the number of computing resources (power, CPU cycles) each computing platform consumes to process the user's task efficiently needs to be ascertained. This paper implements a computational task offloading scheme to a high-performance processor through a small base station and a collaborative edge device through macro base stations, considering the system's maximum available processing capacity as one of the optimization constraints. We minimized the latency and energy consumption, which constitute the system's total cost, by optimizing the computing platform choice, base station's choice, and resource allocation (computing, communication, power). We use the actor-critic architecture to implement a Markov decision process that depends on deep reinforcement learning (DRL) to solve the model's problem. Simulation results showed that our proposed scheme achieves significant long-term rewards in latency and energy costs compared with random search, greedy search, deep Q-learning, and the other schemes that implemented either the local computation or edge computation.

1. Introduction

The wide spread of advanced applications such as interactive online gaming, face recognition, autonomous driving, real-time object recognition, virtual reality (VR), and augmented reality (ARS) comes with a significant challenge in computation with the Internet of things (IoT) devices. This problem is due to the limited processing, memory, and energy resources present in these IoT devices [1–3]. The disadvantages of the IoT device's computational ability have drastically reduced users' quality of experience (QoE) and IoT devices' performance. One way to address this problem is to employ mobile cloud computing (MCC) technology to assist the IoT device's powerful computing and storage capability. However, relying on MCC for the computational offloading service results in the inability to support delay-sensitive applications due to distance from users and data loss, leading to unreliable wireless connection caused by deep fading. Its geographically centralized position caused congestion when there is an explosive growth of users' computational demand [4, 5].

To deal with the drawback seen with MCC, mobile edge computing (MEC) found at the edge of the network is used to

provide computing services [6, 7]. MEC server provides more computational capability than the IoT devices, but its computational capacity is less than MCC because of its distributed structure [6]. MEC enhances the QoE of users due to its proximity to IoT devices. It guarantees traffic decongestion at the core network due to MEC servers' distributed structure and supports delay-sensitive applications due to the transmission latency significantly reduced [6, 8]. Nevertheless, an efficient MEC offloading scheme should still consider if offloading the IoT devices' computational tasks is required due to limited resources. The instantaneous user's available processing capacity (APC) should always be greater than the user's required processing capacity (RPC) needed before deciding to offload [9]. It is important to note that MEC tasks' stochastic nature and frequency indicate that the instantaneous computational demand and input data cannot overhaul to the next scheduling interval. Hence tasks break into units that can be efficiently managed within the scheduling time interval. Moreover, the offloading from IoT devices to MEC servers utilizes the dynamic wireless channel. Therefore, a proper offloading scheme is required to consider the wireless channel's variability too [10–14]. It is usually best to select offloading devices based on their radio link's strength and the critical need for an additional computation facility. Furthermore, power constraints are the major limiting factor to IoT devices' computation capacity, frequently querying the limited MEC servers for computational resources. Hence, to further improve the quality of service (QoS) for IoT devices and reserve MEC for critical requirements, renewable energy harvesters (EH) can be deployed to extend the battery lifetime of IoT devices [15–18]. The IoT device can capture ambient renewable energy such as solar radiation, radio-frequency (RF) signals, wind power generation, and human kinetic motion. These renewable energy sources provide the greener energy required for the IoT central processing unit (CPU) and the radio transceiver using EH module [13, 19, 20].

In [21], the computation of task offloaded from IoT device was implemented using satellite and gateways in satellite communication. Here tasks can be handled by satellite or serve as a medium for task transfer to the gateway where it is computed. When adopted to terrestrial communication, this idea, computing task from IoT devices, can be handled by the high-performance processor connected to the small base station. The small base station also serves as a medium of transfer to a collaborative edge device connected to the macro base station, where the computation of the task is completed. Reference [21] does not consider user computation's effect on overall energy consumption and latency reduction. Also, they did not consider efficiently allocating the available power (CPU and RF modules) to improve the system's available processing capacity. In summary, we considered both the user computation and available power in our overall energy and latency optimization expressed in equation (5), which is an addition to the choice of computing platform, selection of the base station, and resource allocation considered in the work of [21].

The offloading scheme should adapt to the practical scenario whereby IoT devices' tasks are not predictable. Furthermore, due to the MEC tasks' stochastic nature, the computation demand and input data cannot be carried to the next scheduling interval but immediately processed when the task is brought into the network. To achieve this, the user's available processing capacity (APC) at the time instant of task arrival should be greater than the user's required processing capacity (RPC) [9].

However, the APC is a function of the execution latency from [9]. If enhanced, this indeed improves the APC. At [9], the research allocation policy ensures that the user's APC is more than the RPC for computing tasks. In a practical scenario, it means that if the computing task of MEC is constant, for the condition always to hold, then there arise cases where the user's task may not get access into the system due to insufficient power to properly allocate between the CPU and RF modules to make APC more than RPC. In addition, because the computing capacity of the MEC is finite. This problem results in users computing tasks missing their deadline for delay-sensitive tasks. Therefore to support the proposition of APC always greater than RPC, the computing capacity of MEC can be enhanced by adding to its computing ability by forming a D-D connection with all the unengaged devices within the proximity of the MEC. This leads to a better quality of experience (QoE) from the user's perspective. Also, to efficiently utilize the power resources, an exact amount of power needed for the CPU and RF modules for the chosen computing platform for maximizing APC needs to be ascertained as power is a limiting resource. In addition, since there are many deployments of these computing devices that are attached to either the small base station or macro base station, a choice of the particular base station a user can associate from the many available needs to be ascertained that leads to a reduced cost in energy consumption and delay. A resource allocation policy that considers user association, offloading decision, computing platform, and power as a limiting resource with the sole aim of maximizing the APC will not only promote the QoE of users. Still, it further adapts the scheme to the practical scenario of addressing the unpredictable and stochastic arrival of computing tasks into the network. The use of mathematical optimization to solve the following observations is complex. It results in a high cost of implementation and severe delays as the optimization runs at every time slot during the resource allocation policy. The use of deep reinforcement learning leads to a less complicated and low cost of implementation.

Motivated by the observations above, this paper focuses on the MEC IoT networks with multiple users, multiple high-performance processors, and multiple collaborative edge devices. The high-performance processor provides access to the collaborative edge device for tasks offloaded from IoT nodes/users and is not handled by the high-performance processor. The IoT devices are power limited, coupled with their computing and communication resources being scarce. Therefore, considering computing platform choice, base station choice, and resource allocation for task offloading should be investigated to minimize the MEC IoT system's total cost due to latency and energy consumption.

It is quite challenging to solve the formulated optimization problem using the standard techniques due to the

problem's nonconvexity. We point out the primary contributions of this paper as follows:

(i) We demonstrated how the total cost could be optimized considering three-tier computing platforms with multiple users, high-performance processors, and collaborative edge devices. Unlike the existing methods, we considered the joint user association, offloading decision, computation, and communication resources to optimize the system's service latency and energy consumption under a time-varying channel state and optimal APC.

(ii) We increased our system's computing capacity by making it possible for the small base station high-performance processor to assist in processing tasks offloaded from IoT nodes within the coverage of the small base station. Similarly, the collaborative edge device replaced the conventional MEC server seen in other schemes. The task edge device can intelligently form D-D connections with all the unengaged devices (resource edge devices) within its vicinity to complete task execution from the user. It should be noted that tasks handled at the collaborative edge device must pass through the high-performance processor.

(iii) We formulated the problem of minimizing latency and energy consumption while optimizing computing platform choice for execution, the base stations to associate with, and resource allocation. We intelligently determine the appropriate power the computing task requires to maximize the APC. We adopted the actor-critic architecture of deep reinforcement learning proposed in [19, 22] to carry out our investigation.

(iv) We compare our method's performance with other known benchmark algorithms (random search, greedy search, deep Q-learning) together with schemes of local and edge computation and show that our scheme provides a significant reduction in total cost due to latency and energy consumption.

1.1. Related Work. The study conducted by the group known as European Telecommunications Standards Institute (ETSI) first gave the motivation, definition, protocol architecture, and challenging issues of MEC [23, 24]. The computational task offloading mechanism in edge computing is responsible for MEC system all-around performance. The power constraint of IoT devices led to energy-efficient computation offloading, as we see in the study of [23, 25, 26]. In [25], an actual data measurement was suggested based on the optimization method to save users' energy consumption by jointly formulating the joint scheduling and computation problem. Reference [26] proposed the system cost, which considers delay and task failure as a performance metric in the dynamic computation offloading deploying the Lyapunov energy harvesting process. The amount of energy usage and task delay in the MEC method relies only on processing the task and transmitting it. However, based on optimizing radio resources and computing offloading, this consumed energy has increased [13, 27].

Reference [13] investigated the resource allocation for the multiuser MEC offloading problem under TDMA and OFDMA scenarios. Likewise, in [27], joint optimization of computation task scheduling and radio resource allocation was carried out for a multi-access-assisted computing offloading. In all these works, the MEC maintained a static position, unlike the work of [23] in which flying UAVs serve as the mobile edge servers. Also, in [28], they proposed a mobile edge computing framework with the aid of a UAV-mounted cloudlet. They jointly formulated the UAV trajectory and bit allocation problems to reduce mobile energy consumption through a nonconvex optimization solution. The role of edge computing enhanced industrial IoT was carried out in [21, 29, 30]. The nondominated sorting genetic algorithm, as proposed by [31], was used to study the trade-off between the amount of consumed energy and delay for their user-MEC system and further proves that the algorithm can solve problems of similar kinds. A partial and binary offloading of a three-tier computational platform was proposed by [32], which improves the node's energy efficiency factor by optimizing both computing and communication resources. An information asymmetry and information uncertainty as proposed by [33] were used to improve the system service in vehicular fog computing network. In [34], under a user-MEC system, the system performance was enhanced by jointly considering three indices: admission control, power control, and resource allocation. However, the increasing number of IoT devices and small or macro base stations that host the edge computing devices seen with the above schemes must be considered to associate the users appropriately for computational purposes. This consideration is necessary as the number of base stations (small or macro) is generally small compared to the number of IoT nodes.

Reference [35] carried out joint computation and user association for multitasks MEC systems to minimize the overall energy consumption.

Some works focusing on the supply of renewable energy and EH include the following: [19, 36] suggested a security disjoint routing-based verified message (SDRVM) using energy data from Denver's National Solar Radiation Database. Here, solar energy and battery collection for energy storage are included in the energy consumption model. Reference [37] defined the communication between the EH wireless devices (WDs) and energy-transmitting devices by suggesting a wireless powered communication (WPC) model along with a radio-frequency (RF) energy receiver model. Reference [38] minimized capacity using the number of computing bits and causality of energy harvesting as constraints. By proposing a learning-based computing offloading scheme for IoT devices coupled with EH, [1] maximizes system utility.

The delay in data transmission and energy consumption on the wireless network is defined below. Reference [39]

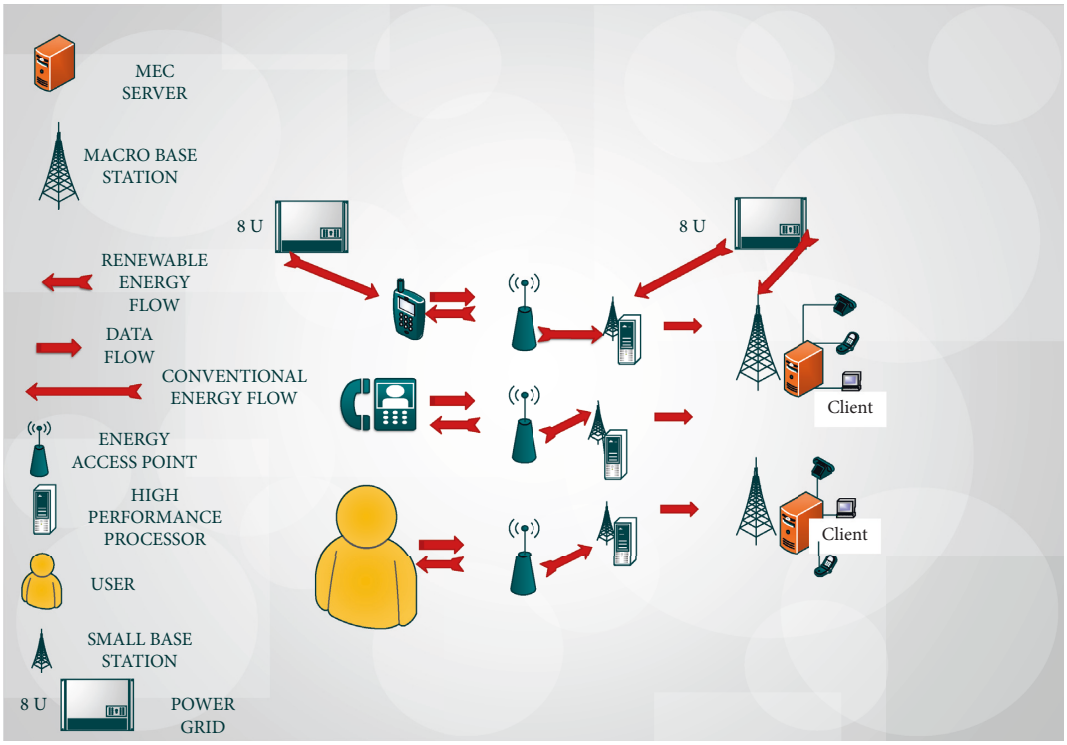

Figure 1: Block diagram of edge-based computation tasks offloading model for IoT device in IoT networks.

improved delay and energy consumption efficiency when changing forwarder nodes and duty cycle using the packet aggregation routing system (AFNDCAR) to monitor node residual energy. The alternating direction method of multiplier (ADMM) to optimize the computation was performed by joint optimization of local computing or offloading to a MEC server and allocating transmission time [14]. To reduce the average offloading delay, they follow an adaptive learning-driven task scheme based on the multi-armed bandit algorithm in [40]. Deep learning (DL) and DRL optimize computing offloading and resource distribution or transmission delay and energy consumption. An example can be seen in the work of [41] which, by implementing a joint offloading and edge server provisioning issue based on RL, minimized the long-term device expense. Using a postdecision state (PDS) learning algorithm, state transformations' unique structure in the MEC system was also implemented.

The actor-critic RL approach was used by [22] to optimize caching, computing, and communication jointly. A two layered RL algorithm is used to find the trade-off between latency and delay for a resource constraint IoT device while offloading computing task to the MEC server as can be seen in [42]. Reference [43] used a three-layer neural network through the design of a DRL-based offloading model to learn the optimal offloading strategy with varying data transmission rates. A DRL-based joint optimization scheme for computing task scheduling and wireless resource allocation in a vehicular network was proposed by [44] by modelling the communication system and edge computing. Likewise, in the work of [45–48] the use of DRL was adopted for an efficient task computation offloading to the edge server to minimize both latency and energy consumption.

2. System Model

A typical MEC IoT network with multiple users, multiple high-performance processors, and numerous collaborative edge devices forming the three computing platforms are presented in Figure 1.

We design a MEC system consisting of three parts where each part is equipped with a computing platform for the task's execution.

(i) IoT nodes layer: This includes the smartphone, visual terminal, etc. Let $\mathcal{N} = \{1, 2 \ldots N\}$ denote a collection of IoT nodes.

(ii) The service equipment layer: This comprises a small base station and high-performance processor. Let $\mathcal{K} = \{1, 2 \ldots K\}$ denote the collection of high-performance processors. Since each high-performance processor is attached to one small base station, it then means we have k high-performance processors and k small base stations in the system.

(iii) The MEC layer: This consists of a macro base station and a collaborative edge device. Let $\mathcal{M} = \{1, 2 \ldots M\}$. Since each collaborative edge device is connected to a macro base station, there are M collaborative edge devices and M macro base stations in the MEC model.

This paper uses the small base station and high-performance processor interchangeably. Similarly, the macro

base station and collaborative edge device refer to the same thing as we have earlier mentioned that the high-performance processor is hosted on a small base station and the collaborative edge device is hosted on a macro base station.

We note that the IoT device and small base station have energy harvesting capability and tap renewable energy from an external source. In contrast, the task edge device subset of collaborative edge devices is powered by a conventional power grid. Thus, the three computing platforms' computing capabilities can be written: IoT node < High-performance processor < collaborative edge device. Data flow from the IoT nodes to the collaborative edge device through the high-performance processor. It means that the task scheduled to be processed at the collaborative edge device passes through the high-performance processor but will not be handled there. We considered a system with multiple IoT nodes, service equipment layers, and MEC layers. The user needs to associate with the best small base station, macro base station, and computing platforms to reduce the cost due to latency and energy consumption. Table 1 lists the key notations mentioned in this work.

2.1. Collaborative Edge Device. It is a set of task edge devices and all the resource edge devices which can form a D-D connection with the task edge device. The connection between the task edge device and resource edge devices is wired. Let us denote a task edge device by t^e and resource edge devices that can establish connection with t^e as $r^e = \{1, 2 \ldots r^e\}$. We have that $t^e + r^e = m$, where m is a collaborative edge device $\in M$.

2.2. Communication Model. Our MEC system adopts OFDMA in the two transmission segments with C, D orthogonal subcarriers represented by $\mathscr{C} = \{1, 2 \ldots C\}$, $\mathscr{D} = \{1, 2 \ldots D\}$, respectively. Let the bandwidth of the small base station and macro base station be denoted as B_1 and B_2, respectively. The uplink transmission rate $R^c_{TX,i}$ for the transmission segment from user ı to small base station k is given by

$$R^c_{TX,i} = \sum_{c \in C} \rho_{i,c} B_1 \log_2\left(1 + P_{i,c} g_{i,c}\right). \tag{1}$$

Similarly, the transmission rate for the second segment from small base station k to the collaborative edge device is given by

$$R^d_{TX,i} = \sum_{d \in D} \rho_{i,d} B_2 \log_2\left(1 + P_{i,d} g_{i,d}\right), \tag{2}$$

where $\rho_{i,c}, \rho_{i,d} \in (0, 1)$ is an indicator variable. $P_{i,c}$ and $P_{i,d}$ represent the power allocated to the subcarriers c and d, respectively. $g_{i,c} = (\|h_{i,c}\|^2 / N_o)$ and $g_{i,d} = (\|h_{i,d}\|^2 / N_o)$ denote their respective channel gains. N_o is the power spectral density of the additive white Gaussian noise (AGWN). h denotes the channel response, while ρ is an indicator variable which is 1 when subcarrier is assigned and 0 when no subcarrier is assigned. To avoid interference, the following indicator variables should be satisfied:

Table 1: Summary of some of the key notations used.

Notation	Description
N	Set of IoT nodes
k	Set of small base stations (high-performance processors)
M	Set of macro base stations (collaborative edge devices)
r^e	Set of resource edge devices
t^e	Task edge device
C	Set of subcarriers from user to small BS
D	Set of subcarriers from small base station to macro BS
B	Bandwidth
$R_{TX,i}$	Transmission rate
$P_{i,c}$	Power allocated to C
$P_{i,d}$	Power allocated to D
ρ	Indicator variable for
$p_{i,0}$	Local computing power
f_i	CPU computational frequency of user ı
ζ	Effective capacitance coefficient
P	The available power
T	Latency
N_i	Number of bits in i
$\varkappa$	Computing capacity of IoT node
z	The computational capacity of small base station
q	Computing capacity of macro base station
V	Propagation latency
T_{eff}	System latency
C_T	Total APC
$C_{\text{req},i}$	Required processing capacity for user i
E_k	Energy consumed at the small base station
$P_{s,k}$	Static power offset
P_k	Total consumed power
r_L	Reward at time slot L
$L(\omega)$	Loss function
α_a	Learning rate
δ_L	Temporal difference
π	Policy
X_k, Y_m	Maximum communication capacity allocated by k and m
Z_k, Q_m	Maximum computing capacity allocated by k and m

$$\begin{aligned} \sum_{i=1}^{N} \rho_{i,c} &\le 1, \\ \sum_{i=1}^{K} \rho_{i,d} &\le 1. \end{aligned} \tag{3}$$

2.3. Computation Model. We assume that the IoT nodes, high-performance processor, and collaborative edge devices can change their CPU computational frequency to adopt the power consumption and execution latency using the dynamic voltage scaling (DVS) techniques [9]. We can model their computational power as follows:

$$p^n_{i,0} = f^n_i \zeta^n_i, \tag{4}$$

$$p^k_{i,0} = f^k_i \zeta^k_i, \tag{5}$$

$$p^m_{i,0} = f^m_i \zeta^m_i, \tag{6}$$

where $p_{i,0}^n, p_{i,0}^k, p_{i,0}^m$ denote the input powers for the IoT device, high-performance processor, and collaborative edge device, respectively. f_i^n, f_i^k, f_i^m denote the CPU computational frequencies measured in Hz of IoT device, high-performance processor, and collaborative edge device, respectively. $\zeta_i^n, \zeta_i^k, \zeta_i^m$ denote their effective capacitance coefficient that depends on the chip architecture. We set $\zeta_i > 0$ for all the computing platforms. We equally set $n = k = m = 2$ with frequency upper bounds given by $f_{\max,i}$. This implies that their computational power satisfies $0 \le P_{i,0} \le f_{\max,i}^2 \zeta_i$. The power present at the first transmission segment is constrained by the available power of the IoT nodes covered by the small base station k and is given by

$$p_{i,0}^n + \sum_{c=1}^{C} p_{i,c} \le P_i^n. \tag{7}$$

Similarly, the second transmission segment is constrained by the available power of the high-performance processor given by

$$p_{i,0}^k + \sum_{d=1}^{D} p_{i,d} \le P_i^k, \tag{8}$$

where $p_{i,0}^n$ and $p_{i,0}^k$ are the CPU modules of the IoT node and high-performance processor, respectively. $\sum_{c=1}^{C} p_{i,c}$ and $\sum_{d=1}^{D} p_{i,d}$ represent the RF modules of the IoT node and high-performance processor (small base station). P_i^n and P_i^k are the available power present in IoT node and high-performance processor, respectively. We ignore the RF modules' received power, consistent with other schemes, as their feedback is negligible.

2.4. Latency Model. For every task i, we can compute the latency in any three computing platforms. This latency includes waiting, transmission, processing, and propagation. We look at the three possibilities in evaluating the system's latency and consider only the processing latency if processed at the IoT node.

2.4.1. Task Handled at the User. For the processing of task, i at the user, we express the latency T_i^n as follows:

$$T_i^n = \frac{N_i}{\varkappa_{i,n}^L}, \tag{9}$$

where N_i is the number of bits in task i at time slot L. $\varkappa_{i,n}^L$ is the computing capacity allocated to task i at time slot L for the IoT node.

2.4.2. Task Handled at the High-Performance Processor. If task i is handled at the high-performance processor $k \in K$, we express the latency T_i^k as follows:

$$T_i^k = \rho(L-1) + \frac{N_i}{C_{i,k}^L} + \frac{N_i}{z_{i,k}^L} + V_{i,k}^L. \tag{10}$$

We use ρ and $\rho(L-1)$ to denote the length of a time slot L and waiting latency. $C_{i,k}^L$ denotes the link's allocated communication capacity from user to high-performance processor, k. $V_{i,k}^L = (2d_{i,k}/S)$ denotes the latency due to propagation for task i processed by high-performance processor k, where $d_{i,k}$ is the distance between IoT node and high-performance processor, k. S is the speed of light. We ignore the transmission delay from small base station k to IoT node n because of the lesser number of bits in the return link but not so for their propagation delay.

2.4.3. Collaborative Edge Device. Task i is processed at the collaborative edge device. It gets offloaded through the small base station to the collaborative edge device without processing at the high-performance processor. Thus we express the latency T_i^m as follows:

$$T_i^m = \rho(L-1) + \frac{N_i}{C_{i,k}^L} + \frac{N_i}{D_{i,m}^L} + \frac{N_i}{q_{i,m}^L} + V_{i,m}^L, \tag{11}$$

where $D_{i,m}^L$ is the communication capacity allocated for the second transmission segment from the small base station to the collaborative edge device. $q_{i,m}^L$ is the allocated computing capacity assigned to the task i by the collaborative edge device. $V_{i,m}^L = (2d_{i,k} + 2d_{i,m}/S)$ where $d_{i,m}$ is the distance between high-performance processor and collaborative edge device. The transmission latency from the collaborative edge device to high-performance processor and from high-performance processor to users is also omitted as seen before.

From equations (9)–(11), the cost due to latency can be defined as

$$T_i = \begin{cases} T_i^n & \text{if } \alpha_i = 1, \\ T_i^k & \text{if } \beta_i = 1, \\ T_i^m & \text{if } \gamma_i = 1. \end{cases} \tag{12}$$

In (12), $\alpha_i, \beta_i, \gamma_i$ are the indicators showing where to offload task i for processing. If $\alpha_i = 1$, task i will be handled at the IoT node, if $\beta_i = 1$, task i goes to the high-performance processor for processing. If $\gamma_i = 1$, task i goes to the collaborative edge device for processing. In addition α_i, β_i, and γ_i must satisfy $\alpha_i + \beta_i + \gamma_i = 1$. Considering the latency of all task i, the total cost of the system due to latency is expressed as

$$T_{\text{eff}} = \sum_{L} \sum_{n} \sum_{k} \sum_{i \in \overline{\Omega}_{k,L}} \omega_i T_i. \tag{13}$$

In (13), we express the cost at time slot L due to latency for our user-MEC system as the summation of weighted latency for all task i scheduled. $\overline{\Omega}_{k,L}$ comprises all the scheduled new tasks that are to be associated with the small base station at time slot L and ω is the weights of each task. With reference to (9)–(11), T_{eff} is related to choice of computing platform, choice of base station, and resource allocation (computing, communication) at each time slot.

3. Available Processing Capacity

The APC of a user ı is given by the maximum available computation it can obtain in the time interval between L and $L + \delta_L$. From [9], it was shown that, at time instant L, user i APC is expressed as

$$C_i = \lim_{\delta_L \longrightarrow 0} \frac{\omega_i}{\delta_L}, \tag{14}$$

where ω_i denotes the maximum available computation obtained within the time interval L to $L + \delta_L$. It was also proved from [9] that APC can be further expressed by

$$\begin{aligned} C_i(P_i, \rho_i) &= \lim_{\delta_L \longrightarrow 0} \frac{\omega_i}{\delta_L} \\ &= \sqrt{P_{i,0}\zeta_i^{-1}} + \eta_i R_{TX,i}. \end{aligned} \tag{15}$$

For any of the unpredictable tasks to be completed without an extra delay, a sufficient condition to realize this is by making the user's APC greater than its required processing capacity (RPC). The RPC is the total computation demands per second of all tasks present at the user. Let us denote the APC of the user from user to the high-performance processor as C_i and that from the high-performance processor to the collaborative edge device to be C_k. The total APC C_T is then given:

$$C_T = C_i + C_k. \tag{16}$$

This implies that, for all unpredictable tasks arriving at the user, $C_T > C_{\text{req}}$ where C_{req} is the required processing capacity at the user.

The APC from user to the high-performance processor (C_i) is given by

$$C_i = \sqrt{p_{i,0}^n \zeta_i^{-1}} + \eta_i R_{TX,i}^c. \tag{17}$$

Similarly the APC from high-performance processor to the collaborative edge device (C_k) is given by

$$C_k = \sqrt{p_{i,0}^k} \zeta_i^{-1} + \eta_i R_{TX,i}^d, \tag{18}$$

where the expression for $p_{i,0}^n$ and $p_{i,0}^k$ has been given in (4) and (5), respectively. Also the expression for $R_{TX,i}^c$ and $R_{TX,i}^d$ has equally been given in (1) and (2), respectively.

We note the following constraints:

(i) (RPC constraint): Here, the resource allocation targets making the total APC C_T of the system satisfy the RPC constraint. We express it as follows:

$$C_T \geq C_{req,i} \forall i \in N \tag{19}$$

(ii) (Server capacity constraint): let Z_k and Q_m be the total processing capacity seen at the high-performance processor and collaborative edge device, respectively. We have that

$$\begin{aligned} \sum_{i \in N} \eta_i R_{\text{TX},i}^c &\leq Z_k, \\ \sum_{i \in K} \eta_i R_{\text{TX},i}^d &\leq Q_m. \end{aligned} \tag{20}$$

3.1. Energy Cost Model. We considered power consumed at the small base station and IoT device as they are the two power-limited devices in our network. We omit power consumed at the macro base station as the power supplied to the static offset power is stable. The energy consumption changes only slightly because the macro base station receives its power from the conventional power grid. Therefore the total cost in energy due to the collaborative edge device hosted on this macro base station is not affected.

According to [49], we can model the power consumed at the small base station k by the sum of its static baseline power and consumed power as follows:

$$E_k(L) = \left[P_{s,k} + \eta P_k(L)\right] * L, \tag{21}$$

where $P_{s,k}$ is the static power offset (baseband processor, cooling system). P_k denotes the total consumed power due to wireless transmission by the small base station k. At time slot (L), we express the components of the total consumed power $P_k(L)$ of the small base station k as

$$P_k(L) = \sum_{n=1}^{N} p_{i,0}^n(L) + \xi_i \sum_{n=1}^{N} p_{i,c}(L) + (1 - \xi_i) p_{i,d}(L), \tag{22}$$

where $p_{i,0}^n(L)$ is the power consumed at the IoT device for local execution at time slot L covered by a small base station k.

$p_{i,c}(L)$ denotes the transmit power of the first transmission segment at time slot L.

$p_{i,d}(L)$ denotes the transmit power of the second transmission segment between the high-performance processor k and collaborative edge device m.

ξ denotes the ratio of offloading to high-performance processor k and the collaborative edge device m.

4. Problem Formulation

In light of the total cost due to latency and energy consumption model seen in the previous section, we formulate our optimization problem to reduce the total cost due to latency and energy consumption at time slot L. Let $\Gamma = (\overline{\Omega}_{m,L}, \overline{\Psi}_{m,L}, \overline{\phi}_{n,L})$. We formulate the problem as follows:

$$\min_{\Gamma, C_{i,k}^L, D_{i,m}^L, z_{i,k}^L, q_{i,m}^L, P_k(L)} \left(\eta T_{\text{eff}} + (1 - \eta) E_k(L), \right. \tag{23a}$$

$$\text{subject to} \sum_{k \in \Omega_{k,L}} C_{i,k}^L \leq X_k, \tag{23b}$$

$$\sum_{k \in \phi_{m,L}} D_{i,m}^L \leq Y_m, \tag{23c}$$

$$\sum_{k \in \psi_{m,L}} z_{i,k}^L \leq Z_k, \tag{23d}$$

$$\sum_{k \in \phi_{m,L}} q_{i,m}^L \leq Q_m, \tag{23e}$$

$$P_{i,0}^n + \sum_{c=1}^{C} p_{i,c} \leq P_i^n, \quad \forall i \in N, \tag{23f}$$

$$P_{i,0}^{k} + \sum_{d=1}^{D} P_{i,d} \leq P_{i}^{k}, \quad \forall i \in K, \tag{23g}$$

$$\sum_{i=1}^{C} \eta_i R_{TX,i}^{c} \leq Z_k, \quad \forall i \in N, \tag{23h}$$

$$\sum_{i=1}^{D} \eta_i R_{TX,i}^{d} \leq Q_m, \quad \forall i \in K, \tag{23i}$$

$$C_T \geq C_{\text{req},i}, \quad \forall i \in k, \tag{23j}$$

$$0 \leq P_{i,0} \leq f_{\max^2} \zeta_i, \tag{23k}$$

$$p_{i,c}, p_{i,d} \geq 0. \tag{23l}$$

$\Omega_{k,L}$ denotes at time slot L the tasks associated with a high-performance processor (small base station) k. It consists of the ongoing tasks and new tasks $\overline{\Omega}_{k,L}$ scheduled to be processed at time slot L at the high-performance processor k. $\phi_{m,L}$ represents all the tasks to be processed at the collaborative edge device m at time slot L and $\cup_m \phi_{m,L} \subset \cup_k \Omega_{k,L}$ because the tasks offload to the collaborative edge device through the small base station (high-performance processor) at time slot L. $\psi_{k,L}$ consists of all the tasks scheduled to be processed at the high-performance processor k at time slot L. $\psi_{k,L} \subset \Omega_{k,L}$ because some of the new tasks are not processed at the high-performance but merely pass through there to the collaborative edge device where they are processed. X_k and Y_m are the highest available communication capacity available for the transmission segment from IoT device/users to the high-performance processor and high-performance processor to the collaborative edge device, respectively. Similarly, Z_k and Q_m are the highest available computing capacity available at the high-performance processor k and collaborative edge device m, respectively. We also point out that all the resources deployed for the ongoing tasks $i \in (\Omega_{k,L}/\overline{\Omega}_{k,L})$ can no longer be utilized for the new task scheduled at time slot L. η and $1 - \eta$, respectively, represent the latency weight due to latency and weight due to energy consumption. It should satisfy $0 \leq \eta \leq 1$. Constraint (23b) signifies that the summation of all the communication resources (RF modules) allocated to the set of users connected to the small base station should not be more than the maximum communication capacity the high-performance processor can offer. Similarly, constraint (23c) indicates that the summation of all the communication resources from the high-performance processor to the macro base station should not be more than the maximum communication capacity present at the collaborative edge device. Constraint (23d) signifies that the total computation offloaded to the small base station should not be more than the high-performance processor's processing capacity. Constraint (23e) also notes that the total computation offloaded to the macro base station should not exceed the collaborative edge device's processing capacity. Constraints (23f) and (23g) show that the sum of the computation and communication powers should not exceed their available powers. Constraints (23h) and (23i) show that the sum of the RF modules for all the users offloaded to the small base station should not exceed the high-performance processor's computing capability. It is also similar to (23i) for the sum of RF modules offloaded to the macro base station. Constraints (23j) make the user's total APC be more than the required processing capacity of a user i. Constraints (23k) and (23l) indicate that there should be positive power allocation and that the computational power has a maximum limit.

We note from 25 that the total cost due to latency and energy consumption depends on the choice of computing platform, choice of base station users can associate, and resource allocation at each time slot L. Also, the three metrics coupled together at each time slot $L + 1$ depend on the state of time slot L. Hence, we formulate the problem as a dynamic programming problem based on metrics considered. We deploy actor-critic architecture based on deep reinforcement learning to solve the problem since it involves many variables.

4.1. Latency and Energy Optimization Based Deep Reinforcement Learning. The optimal solution to the problem formulated in this paper is a mixed-integer problem of nonconvex nature. It is hard to get a solution due to a combination of discrete and continuous variables. The CPU modules and RF modules that make up the available power for each computing platform and resource allocation variables are continuous. In contrast, the user association and offloading decision variables are discrete. To reduce the complexity in finding a solution, we use an actor-critic DRL-based algorithm to solve the joint user association, offloading decision, and resource allocation that involves many states and action space. We represent the MDP using (H, A, P, R), where H is the system state space, A represents the system action space, and P denotes the transition probability space from state h_L and action a_L.

4.1.1. State (H). The system states are expressed for all time slots L since all the parameters we want to optimize are defined therein. Hence, we represent the state at time slot L as follows:

$$h_L = \left\{\widetilde{\Omega}_L, \widetilde{\psi}_L, \widetilde{\phi}_L, \widetilde{U}_L, \widetilde{X}_L, \widetilde{Y}_L, \widetilde{Q}_L, \widetilde{Z}_L, \widetilde{\sigma}_L, \gamma(L), d_k, M, N,\right\} \tag{24}$$

$\widetilde{\Omega}_L$ is further expressed as $\left\{\widetilde{\Omega}_{1,L}, \widetilde{\Omega}_{2,L} \ldots \widetilde{\Omega}_{k,L}\right\}\widetilde{\Omega}_{k,L}$ is a collection of ongoing tasks connected to the high-performance processor (small base station) k at time slot L.

$\widetilde{\psi}_L$ is further expressed as $\left\{\widetilde{\psi}_{1,L}, \widetilde{\psi}_{2,L} \ldots \widetilde{\psi}_{k,L}\right\}\widetilde{\psi}_{k,L}$ is a collection of ongoing tasks being computed at the high-performance processor k time slot L.

$\widetilde{\phi}_L$ is further expressed as $\left\{\widetilde{\phi}_{1,L}, \widetilde{\phi}_{2,L} \ldots \widetilde{\phi}_{m,L}\right\}\widetilde{\phi}_{m,L}$ is a collection of ongoing tasks at time slot L being computed at the collaborative edge device n.

$\widetilde{X}_L$ is further expressed as $\left\{\widetilde{X}_{1,L}, \widetilde{X}_{2,L} \ldots \widetilde{X}_{k,L}\right\}\widetilde{X}_{k,L}$ is a collection of communication resources at time slot L engaged from the users to the high-performance processor.

$\widetilde{Y}_L$ is further expressed as $\{\widetilde{Y}_{1,L}, \widetilde{Y}_{2,L} \ldots \widetilde{Y}_{m,L}\}\widetilde{Y}_{m,L}$ is a collection of communication resources at time slot L engaged from high-performance processor to the collaborative edge device.

$\widetilde{F}_L$ denotes the collection of tasks at time slot L not scheduled (served).

$\widetilde{Q}_L$ is further expressed as $\{\widetilde{Q}_{1,L}, \widetilde{Q}_{2,L} \ldots \widetilde{Q}_{k,L}\}\widetilde{Q}_{k,L}$ is the set of computing resources at time slot L allocated by the high-performance processor to the ongoing task.

$\widetilde{Z}_L$ is further expressed as $\{\widetilde{Z}_{1,L}, \widetilde{Z}_{2,L} \ldots \widetilde{Z}_{m,L}\}\widetilde{Z}_{m,L}$ is the collection of computing resources at time slot L allocated by the collaborative edge device to the ongoing task.

σ_L represents the matrix that shows the location of all IoT nodes, high-performance processor k, and collaborative edge device m.

γ is a vector list of SINR of the transmission segment between users and high-performance processor and a list of SINR of the transmission segment between high-performance processor and task edge device.

d_k is the computational requirement for task k.

K is the number of high-performance processor.

M is the number of task edge devices.

4.1.2. Action (A). The unserved tasks $\in \widetilde{f}_L$ need to be associated with the small base station k and collaborative edge server m at time slot L. Also, resources for computing and communication need to be assigned to the appropriate choice of the computing platform and base station.

We define at time slot L the action space for the choice of computing platform, choice of base station user can associate, and resource allocation (computing and communication) of the system as follows:

$$\begin{aligned} a_L &= \{\overline{A}_{1,L}, \overline{A}_{2,L} \ldots \overline{A}_{k,L}\}, \\ \overline{A}_{k,L} &= \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8\}. \end{aligned} \tag{25}$$

$A_1 \in 0, 1$ denotes whether an IoT device is capable or not of completing a task at time slot L.

$A_2 \in 0, 1$ depicts at time slot L if task i should be considered for processing.

$A_3 \in 1, 2 \ldots M$ denotes at time slot L the small base station that associates with the task i.

$A_4 \in 0, 1$ indicates in which of the computing platforms the task should be processed.

$A_5 \in 1, 2 \ldots n$ denotes the collaborative edge device used to handle the task k.

$A_6 = \{E_{1,0}, E_{2,0} \ldots E_{i,0} \ldots E_{N,0}\}$ where $E_{i,0}(L)$ can be obtained by $P^n_{i,0} * L$.

$A_7 = \{E_{1,c}(L), E_{2,c}(L), \ldots E_{i,c}(L), \ldots E_{K,C}(L)\}$ where $E_{i,c}(L)$ can be obtained by $P_{i,c} * L$.

$A_8 = \{E_{1,d}, E_{2,d}, \ldots E_{i,d} \ldots E_{M,D}\}$ where $E_{i,d}$ can be obtained by $p_{i,d} * L$.

A_9 = number of communication and computing resources allocated to users from high-performance processor denoted as $C^L = \{C^L_{1,k}, C^L_{2,2} \ldots C^L_{i,k}\}$ and $Z^L = \{Z^L_{1,k}, Z^L_{2,k} \ldots Z^L_{i,k}\}$, respectively, at time slot L.

A_{10} = number of communication and computing resources allocated by the collaborative edge device to tasks offloaded from high-performance processor denoted as $D^L = \{D^L_{1,k}, D^L_{2,k} \ldots D^L_{i,k}\}$ and $q^L = \{q^L_{1,k}, q^L_{2,k} \ldots q^L_{i,k}\}$, respectively, at time slot L. Under a particular action a_L, we obtain the choice of computing platform, base stations to which users can associate $(\overline{\Omega}_L, \overline{\psi}_L, \overline{\phi}_L)$, and appropriate computing and communication resources $(C, D, Z, q, E_{i,0}, E_{i,c}, E_{i,d})$.

4.1.3. Transition Probability (P). A model-free DRL framework is adopted here for the following reasons: It is hard to get the MDP transition probability from one state to another with an action a_L because some state has continuous variables. Secondly, it involves large state space and action space.

4.1.4. Reward (R). We define in time slot L the reward r_L under state h_L and action a_L that minimizes the system weighted cost due to latency and energy consumption as follows:

$$r_L = \sum_u \sum_k \left(\sum_{i \in \overline{\Omega}_{k,L}} \left[j - \eta \omega_i T_i + (1 - \eta) E_{k,L} \right] + \sum_{i \in \overline{F} / \left(\sum_k \overline{\Omega}_{i,L} \right)} -(\eta \omega_i T)_i \right). \tag{26}$$

From (26), the state h_L and action a_L will affect the reward r_L. The reward $j - (\eta \omega_i T_i + (1 - \eta) E_{k,L})$ that is negatively correlated with the weighted-sum of latency and energy cost is fed back for every new scheduled task $i \in \overline{\Omega}_{k,L}$ at each time slot L, whereas the reward $-(\eta \omega_i T_i)$ is fed back for tasks $i \in (\overline{F}_L / (\sum_k \overline{\Omega}_{k,L}))$ in which only the waiting latency constitutes the value of T_i and the value of $E_{i,L}$ is zero due to the knowledge that the task has not been scheduled.

4.1.5. Policy and Value Function. We use the stochastic policy to optimize the long-term performance of the action selection strategy as $\pi(a|s) = \Pr(a_L = a | s_L = s)$. We equally define the expected return of a trajectory that begins at time L under state s and action a as

$$Q^\pi(s, a) = E\left\{ \sum_{i=0}^{\infty} \beta^k r_{L+i} | \pi, s_t = S, a_L = a \right\}, \tag{27}$$

(1) Require: Data from the user/IoT nodes $(N_i, \widetilde{U}_L)$
Data from the high-performance processor $(\widetilde{\Omega}_L, \widetilde{\psi}_L, \widetilde{X}_L, \widetilde{Q}_L)$ data from the Task edge device $(\widetilde{\phi}_L, \widetilde{Y}_L, \widetilde{Z}_L)$ position of the nodes (σ_L) choice of computing platform $(\widetilde{\Omega}_L, \widetilde{\psi}_L, \widetilde{\phi}_L)$ computing and communication resources (C^L, D^L, Z^L, Q^L, E)
(2) Network initialization
Initialize parameters of the actor and critic network $(\theta, \theta^t, \omega, \omega^t)$
(3) For incident = 1 to $E_{\max}$ perform
(4) Renew the environmental situation of the proposed user-collaborative edge device model
(5) reset the state S_o
(6) reset $r_t = 0$
(7) for step = 1 to $T_{\max}$ do
(8) Choose a_t in the simulation environment in line with $\pi\theta(a|s)$
(9) obtain the reward r_t and the next state S_{t+1}
(10) cache $\langle S_t, a_t, r_t, S_{t+1}\rangle$ in the replay buffer D as the experiences used for training the actor and critic network
(11) Arbitrarily extract minibatch of I turples from D that will be utilized for training the primary network of the actor and critic
(12) Update the critic network parameters as follows: $w \leftarrow w + \alpha_c (1/I) \sum_{i=1}^{I} [Q_t^i - Q_w(S_t^i, a_t^i)] \nabla_w Q_w(S_t^i, a_t^i)$
(13) Update the actor-network parameters as follows: $\theta \leftarrow \theta + \alpha_a (1/I) \sum_{i=1}^{I} \nabla_\theta^N at\, j(\pi\theta)$
(14) change the two target networks parameters every X steps as follows: $W^t \leftarrow \tau W + (1-\tau) W^t$, $\theta^t \leftarrow \tau\theta + (1-\tau)\theta^t$. Where $\tau = 0.001$
(15) end for
(16) end For

Algorithm 1: Optimal resource allocation for the user-collaborative edge device

where $\beta \in (0, 1)$ is a discount factor.

By selecting the greedy action, the policy for estimating the Q-values for all state and action pair (s, a) can be derived as follows:

$$\pi(a|s) = \arg\max_a Q^\pi(s, a). \tag{28}$$

4.1.6. Evaluating Value Function. The value function $Q^\pi(s, a)$ can be denoted as $Q_w(s, a)$ making use of a fully connected DNN with many hidden layers which are parameterized by collection of weights $W = \{w_1, w_2 \ldots w_n\}$. To realize this, the two units of the DNN input layer introduce to the hidden layers the system state s and action a. We express the outcome of the jth neuron located in layer i, which makes use of ReLu as the nonlinear activation function as

$$y_{ij} = \max\left\{0, \left(w_i.x_i + b_{ij}\right)\right\}, \tag{29}$$

where y_{ij} denotes the output value, x_i denotes the inputs for layer i, w_i denotes the associated weights for the neuron inputs, and b_{ij} is a bias. The estimated Q-value $Q_w(s, a)$ is provided by the output layer of DNN. By repeatedly reducing the loss function, the DNN acquires knowledge of the best fitting weight as follows [22]:

$$L(w) = E\left[r_L + \beta \max_{a_{L+1}} Q_w(s_{L+1}, a_{L+1}) - Q_w(s_L, a_L)\right]^2, \tag{30}$$

where w represents the parameters of the neural network while the target value is the term $r_L + \beta \max_{a_{L+1}} Q_w(s_{L+1}, a_{L+1})$. The difference between the target value from the estimated value gives us the error also known as temporal-difference (TD) error, δ_L given as

$$\delta_L = r_L + \beta \max_{a_{L+1}} Q_w(s_{L+1}, a_{L+1}) - Q_w(s_L, a_L). \tag{31}$$

4.1.7. The Actor-Critic Architecture of the Deep RL Method to Finding a Solution. The actor-critic algorithm shown in Algorithm 1 was used to combine the concept of policy-based and value-based methods by estimating collections of the actor and critic parameters at the same time [22, 50]. The framework for the actor-critic framework shown in Figure 2 comprises actors that map states to actions and critic that maps state-action pairs to expected long-term cumulative reward [22]. DNN is used in both actor and critic networks to correctly predict value function and policy due to the accurate prediction it provides [22]. The actor uses a parameterized stochastic policy $\pi\theta(a|s)$ to generate an action after observing the current state of the environment. The critic then evaluates action performed by the actor by TD error (or loss function signal) resulting from estimating Q-value with $Q_w(s, a)$ after observing the reward and the next state of the environment. The output from the critic guides the learning seen in the network of the actor and critic. The teaching perfects when the actor uses the critic's output to either increase or decrease its action probabilities. The actor increases its action probabilities if the critic's outcome showed good performance and decreases its action probabilities if it was terrible. Similarly, the critic network parameters are updated to improve the predicted Q-value using the gradient descent method. However, the actor and critic DNNs are trained using experience buffer D.

4.1.8. Critic DNN. One of the problems with this architecture is the inability to converge easily. To facilitate this convergence issue and improve the stability of the algorithm, we employ fixed target network techniques [51] and replay of experience

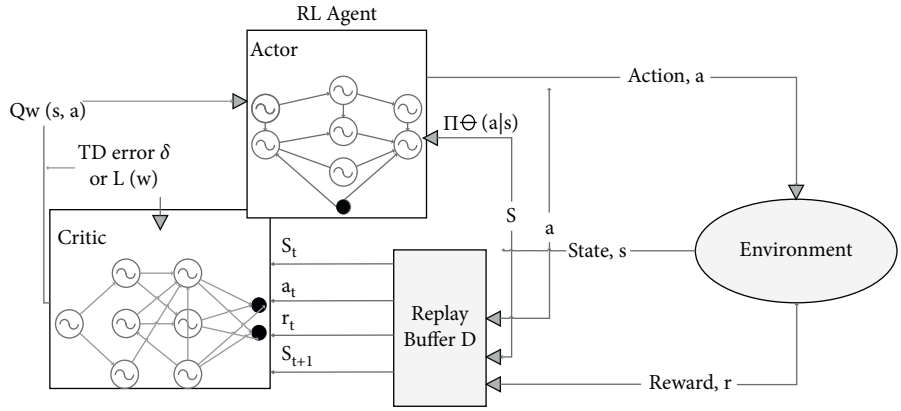

Figure 2: The actor-critic learning framework.

[52] to remove problems of nonstationary targets and break up the temporal correlations arising from the different training episodes [22]. The replay buffer stores experiences in the form of I turples as$\langle s_L, a_L, r_L, s_{L+1}\rangle$ and provides$I$minibatches which are sampled to allow DNN update parameters W.

Using the fixed target network and experience replay techniques, we get the loss function which can be expressed as

$$L(w) = E_D\left[r_L + \beta \max_{a_{L+1}} Q_{w^L}(s_{L+1}, a_{L+1}) - Q_w(s_L, a_L)\right]^2, \tag{32}$$

where w^L and D are the parameters of the target network and experience replay buffer, respectively. If we continuously differentiate the loss function with respect to parameter w, the gradient of loss function updates the parameters w of the critic DNN as follows:

$$\triangle w = \alpha_c\left[Q_L - Q_w(s_L, a_L)\right]\nabla_w Q_w(s_L, a_L), \tag{33}$$

where $Q_L = r_L + \beta \max_{a_{L+1}} Q_{w^L}(s_{L+1}, a_{L+1})$ represent the target value and α_c denotes learning rate of the critic. We use the average value over the minibatch to update parameters w as follows:

$$\triangle_w = \alpha_c \frac{1}{l}\sum_{i=1}^{l}\left[Q_L^i - Q_w\left(s_L^i, a_L^i\right)\right]\nabla_w Q_w\left(s_L^i, a_L^i\right), \tag{34}$$

where Q_L^i can be gotten from all critic network output and immediate reward.

4.1.9. Actor DNN. The agent observes most of the training samples that have good rewards due to taking the right action and then tries to improve the probabilities of selecting those right actions. Similarly, it gives negative rewards when poor actions are taken and does not increase the probability of favouring those wrong actions. We make use of a different parameterized DNN to depict the network of the actor, and the initial policy depicts arbitrarily collection of parameters $\theta = \{\theta_1, \theta_2 \ldots \theta_n\}$. There is one unit from the actor's DNN input layer to pass information to the hidden layer of their current state. The actor DNN output layer then decides the current system state. There is experience replay also in the actor-network to cache samples of minibatch used to train the DNN. The policy objective function iteratively improves the policy to maximize the long-term average reward as follows:

$$\begin{aligned} j(\pi\theta) &= E\{Q^\pi(s, a)\} \\ &= \sum_S d(s)\sum_A \pi\theta(a|s)Q^\pi(s, a), \end{aligned} \tag{35}$$

where $d(s)$ denotes the state distribution. When the objective function is partially differentiated with parameters θ, we get the following gradients:

$$\nabla_\theta j(\pi\theta) = \sum_S d(s)\sum_A \nabla_\theta \pi\theta(a|s)Q^\pi(s, a). \tag{36}$$

When approximated Q-value $Q_w(s, a)$ is used, the partial differentiation of the objective function with respect to parameter θ gives the approximated gradient as follows:

$$\nabla_\theta j(\pi\theta) = \sum_S d(s)\sum_A \nabla_\theta \pi\theta(a|s)Q_w(s, a). \tag{37}$$

The natural policy-gradient method is adopted to avoid standard methods that seldom converge to the local maximum with high variance. The natural policy-gradient process looks for the steepest ascent direction concerning the fisher information metric (FIM) given in [53] which is expressed as

$$F(\theta) = \sum_S d(s)\sum_A \pi\theta(a|s)\nabla_\theta \pi\theta(a|s)\nabla_\theta \ln\pi\theta(a|s)^T. \tag{38}$$

We get the natural gradient of the policy by using the inverse FIM to transform the standard gradient as follows:

$$\nabla_\theta^{Nat} j(\pi\theta) = F^{-1}(\theta)\nabla_\theta j(\pi\theta). \tag{39}$$

The parameter θ is updated towards the natural gradient with respect to the learning rate α_a as follows:

$$\nabla_\theta = \alpha_a \nabla_\theta^{N\,\mathrm{at}} j(\pi\theta). \tag{40}$$

5. Numerical Results

This section evaluates the performance of our proposed joint optimization of computing and communication resources,

choice of computing platform, and selection of base station users can associated with minimizing the total cost due to latency and energy consumption of our MEC system through computer simulations.

5.1. Simulation Setup. We considered a framework of $n \in N$ IoT devices, $k \in K$ high-performance processor, and $m \in M$ collaborative edge devices. We assumed that each IoT device is at an equal distance of 20 m from the small base station and 80 m from the macro base station. At a particular time slot L, each small base station can cover one IoT device. Similarly, the macro base station can cover one high-performance processor at each time slot. Moreover, $L = 2$ ms. The maximum allocated power for IoT devices $P^{n}_{i,0,\max}$ is 0.5 W. The maximum allocated transmit power of an IoT device to high-performance processor, k, is $P^{\max}_{i,c}$ of 0.5 W. Also the maximum transmit power from the small base station k to the collaborative edge device m is set to be $p^{\max}_{i,d}$ of 0.5 W. The maximum power at the collaborative edge device $P^{m}_{i,0,\max}$ is 0.5 W. We equally set each small base station's static power $P_{s,k}$ to be 0.5 W. We set the maximum communication and computing capacity of the high-performance processor to be $X_k = 100$ Mbps and $Z_k = 10^{10}$ cycles/s respectively. We set the maximum communication and computing capacity of the collaborative edge device to be 500 Mbps and 5×10^{10} (cycles/s). The maximum bandwidth of the collaborative edge device and high-performance processor is 6 MHz and 3 MHz.

We used a fully connected DNN with two hidden layers of 300 neurons and a ReLu activation function. Three hundred neurons were used in the hidden layers to maintain a trade-off between higher complexity in computation and accuracy for the value function approximation. Too many neurons lead to higher complexity in analysis. We generate two separate target networks for the actor and critic to regularize the learning algorithm and increase stability. It is followed by replacing the actor and critic network's parameters once every $X = 300$ iteration using the current estimates of their primary network's parameters when $\tau = 0.001$. We use an experience replay buffer of size 10000 for training the DNN that returns a minibatch of experiences of 64 when needed. We set the full episode and the maximum number of steps in each episode to 1000. The learning rate of the actor α_a and critic α_c is, respectively, 0.0001 and 0.001. We showed the other scenario and actor-critic DRL parameter in Tables 2 and 3, respectively.

5.1.1. Convergence of the Proposed Scheme. In this subsection, we showed the training results of the proposed scheme. The convergence process of $L(\theta)$, as defined in the training process, is shown in Figure 3.

Here we see that $L(\theta)$ converges at about 1.1×10^4 steps, which implies that the approximated Q-function $Q(h, a, \theta)$ is close to the target action-value function $Q^*(h, a)$. The complexity of the convergence steps due to the large action space in our work reduces. This reduction is because our proposed scheme deals with the large action space during the training process and the offloading, and we obtain resource allocation on the trained Q-network with no iterations. Similarly, total cost defined in (4) converges almost synchronously with $L(\theta)$ at about $1.05 \times 10^4 th$ episodes shown in Figure 4.

We can conclude that the algorithm not only converged, but it did so at a faster rate making it suitable for a dynamic environment. Our model's convergence analysis went like this: The gradient sharing operation facilitated exploration in our approach, which automatically resulted in various policies among agents. This policy diversification opens up a more extensive search field without jeopardizing long-term convergence to the best policy. In the same way, there is less weight in entropy loss terms for long-term convergence to the optimal policy that does not jeopardize exploration.

Table 2: Simulation parameters.

Scenario parameters	Value
Task size N_k	$8 \times 10^4 - 1.2 \times 10^5 bit$
Weight of each task ω_k	1
X_k	10 Mbps
Y_m	50 Mbps
Z_k	10^10 cycles/s
Q_m	5×10^10 cycles/s

Table 3: Actor-critic DRL parameters.

Description	Value
Entropy weight	0.005
Clip value of the gradient clipping	40.0
Buffer size	10,000
Minibatch size of DNN	64
Maximum episode	1000
Maximum number of steps in each episode	1000
Actor learning rate α_a	0.0001
Critic learning rate α_c	0.001
Number of iterations X	300
Activation function of DNN	ReLu
Number of hidden layers of DNN	2
Number of neurons in the hidden layers	300

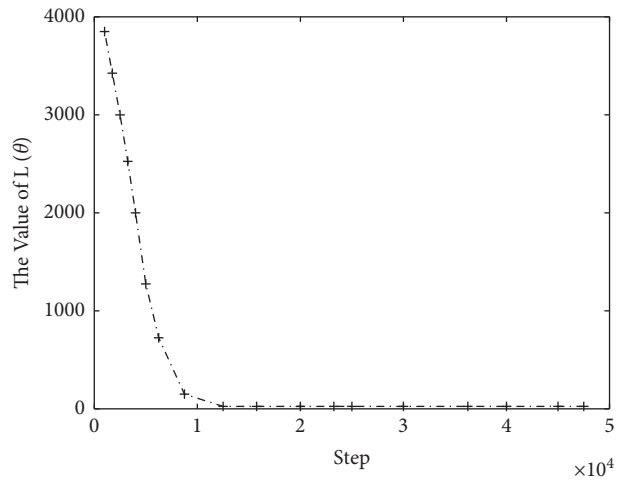

Figure 3: Convergence process of loss.

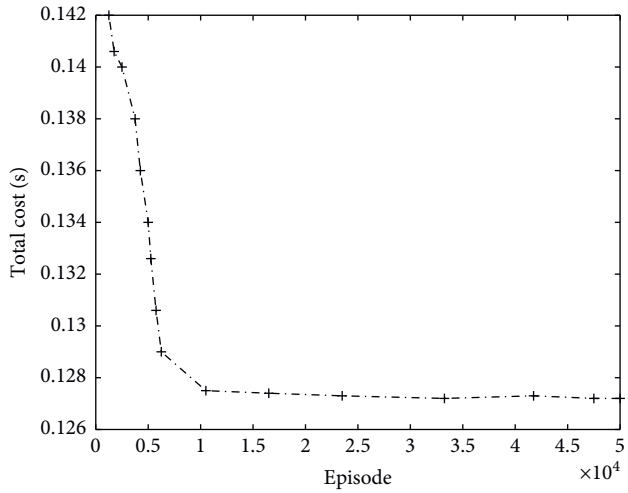

Figure 4: Convergence process of total cost.

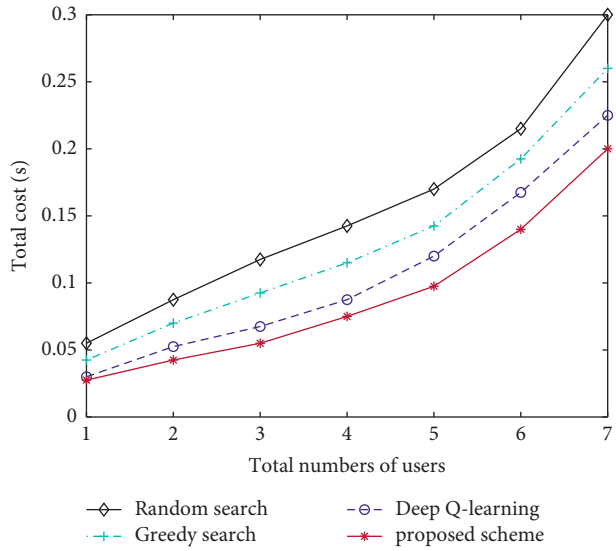

Figure 5: Total cost versus the total number of users.

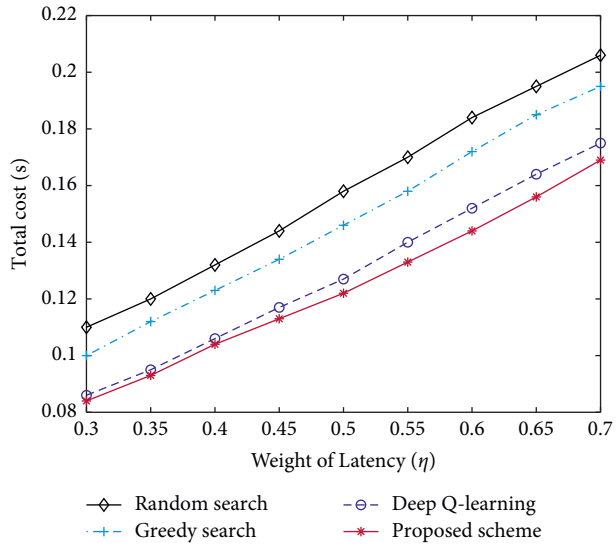

Figure 6: Total cost versus weight of latency (η).

Furthermore, the algorithm's search space was extended by the diverse rules from the gradient sharing operation, increasing the chances of finding the rewarding path. It is not easy to achieve long-term convergence if we do not regulate the sharing process. This regularization was achieved by ensuring that the process complies with [54] work by selecting an appropriate learning rate and clipping gradients. The algorithm's initial investigation was more thorough because of the various policies, but it automatically converged to the optimal policy in the long run due to the lower bias. We also ran the other performance test, seeing the converged algorithm, which measured average service latency and average service energy usage.

5.1.2. Performance Analysis. In the MEC system, we compare our scheme's performance with three benchmark algorithms, which are random search, greedy search, and deep Q-learning.

We take an average of over 5000 tests for every point in the simulation results. From Figure 5, we found that our proposed algorithm performed better in the system's total cost than the other benchmark algorithms. In terms of this total cost, random search increased from 0.055 s to 0.3 s when users increased from 1 to 7. Greedy search increased from 0.035 s to 0.25 s. Deep Q-learning increased from 0.030 s to 0.225 s and finally our proposed scheme increased from 0.0275 s to 0.2 s. These showed that our proposed scheme improved the total cost by 42.03%, 24.64%, and 13.04% when compared with random search, greedy search, and deep Q-learning, respectively. The rise in total cost results from sharing the limited resources to all the participating users in a time slot L. However, our scheme can intelligently deploy more of the task computation at the users when there is competition in the computing and communication resources at the edge servers. Also, users' renewable energy improves their lives to complete tasks when there is no appropriate edge device the user can associate.

Figure 6 shows the dependence on the total cost on weight of latency and energy consumption. The plots show a linear relationship between the total cost and weight. For the delay-sensitive task, a higher value of importance η is required to prioritize delay over energy consumption for the execution in any of the three computing platforms. Our scheme performed better than the other one because it associates the task with the best small base station or macro base station for faster computation.

In Figure 7, we saw that the increase in the number of small base stations decreases the system's total cost, which is the same for all the schemes. As the number of small base stations varies, our method showed better performance because it associates their computing task to the appropriate small base station for processing. Our scheme considers the energy expended at the small base station and intelligently allocates the computing and communication resources in a way that leads to minimizing energy consumption, which results in an overall reduction in total cost. We considered

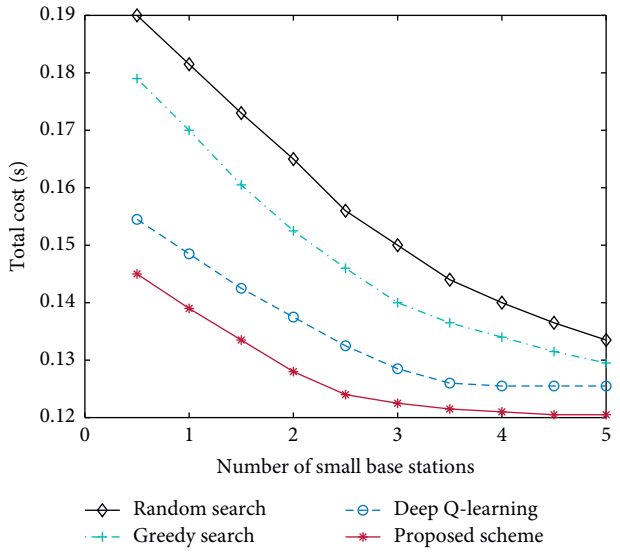

Figure 7: Total cost versus number of small base station.

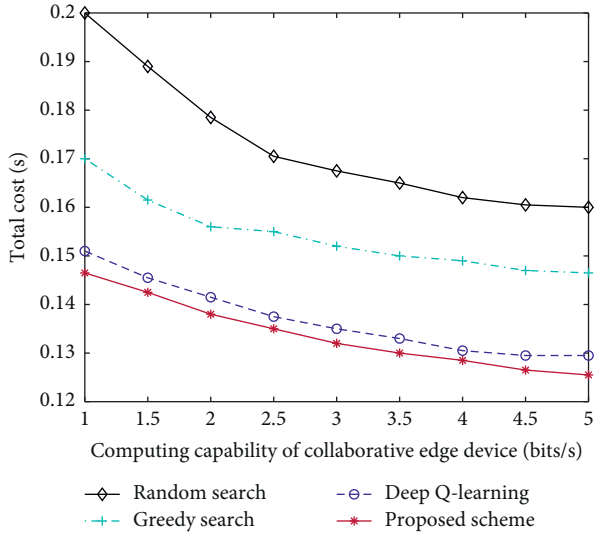

Figure 8: Total cost versus collaborative edge device computing capability.

the small base station's energy consumption since IoT nodes receive their power from the small base station's coverage for its local computation and offloading to the high-performance processor. The available power present in the small base station also provides the transmit power to the collaborative edge device.

Similarly, the graph of the total cost for the collaborative edge device computing capability shown in Figure 8 shows our scheme performed better than the other methods when the collaborative edge device's computing capabilities increased. Our collaborative edge device consists of one task edge device and resource devices. The capabilities of our collaborative edge device are varied by the number of resource devices connected to it. The edge device computing capability is not fixed as in the other schemes. Our system's task edge device subset of the collaborative edge device

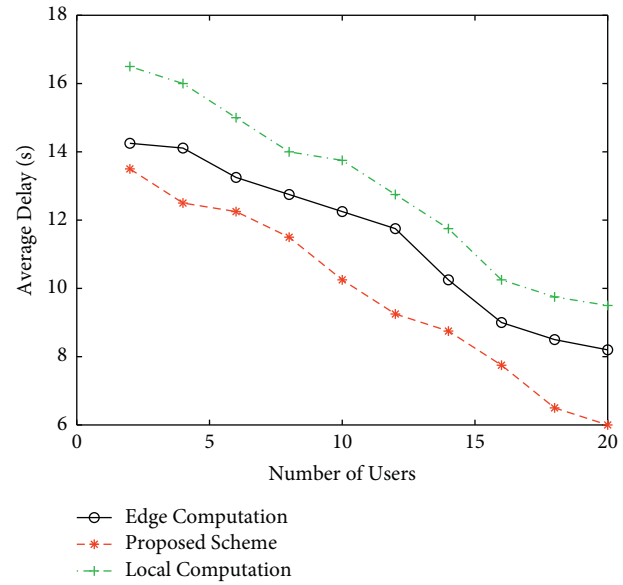

Figure 9: Average delay versus number of users.

adequately uses all those not engaged for computation. These resource devices increase their computing ability and complement the analysis of tasks not handled at the high-performance processor.

We also saw that deep Q-learning and our proposed scheme performed optimally in total cost when the edge servers' computing capabilities were varied. The total cost reduction is higher with these two schemes because the offloading rate is higher than the greedy search and random search with a low offloading rate. However, our proposed system still had the best-reduced cost due to the high-performance processor's presence that had assisted with the task computation before being offloaded to the collaborative edge device. The small base station boosts the user's transmit power for its onward journey to the collaborative edge device, contributing to our scheme's increased offloading rate.

The average delay of the proposed scheme, local computation, and edge computation decreases as the number of users varies from 2 to 20, as shown in Figure 9. The average delay of the local computation showed a minor performance due to the low processing power of the IoT nodes. This delay in computation increases the average delay as tasks get queued waiting to be scheduled for computation. Unlike the schemes that involve local computation, the edge computation and the proposed scheme showed lesser average delay as the number of users increased due to the higher computing power of the MEC to compute tasks at a faster rate. However, our scheme showed the best performance in the average delay as the capacity of the MEC is further improved to handle more complex applications by forming a D-D connection with all the unengaged devices within its vicinity.

6. Conclusion

This paper has shown an effective use of the three computing platforms for processing computationally intensive tasks.

Our proposed algorithm chooses the best computing platform and optimally allocates computing and communication resources to process task. The high-performance processor between the user and the collaborative edge device increases our system's available processing capacity to adapt to the practical scenario where task arrival in the system is random and unpredictable. Our approach ensures that the appropriate computing resources are attained to increase the available processing capacity. It also ensures that our system can reliably handle complex and unpredictable computing tasks that arrive at the network by ensuring that available processing capacity (APC) is always more significant than the required processing capacity (RPC). We employ actor-critic deep reinforcement learning to solve the joint problem of allocating resources and deciding on the best computing platform. Simulation results showed a minimized cost of latency and energy consumption adopting our proposed scheme. In furthering this work, the priority of the user's task to access the network to be processed should be noted. The maximal APC obtained in each scheduling time should consider high-demanding applications and user's satisfaction.

Data Availability

The dataset used to support this article's conclusion is available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for iot devices with energy harvesting," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1930–1941, 2019.

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

[3] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: a viewpoint of vehicles as the infrastructures," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, 2016.

[4] S. Tong, Y. Liu, M. Cheriet, M. Kadoch, and B. Shen, "Ucaa: user-centric user association and resource allocation in fog computing networks," *IEEE Access*, vol. 8, Article ID 10671, 2020.

[5] C. S. Chidume, O. Nnamani, A. Ajibo, and J. M. M. C. I. Ani, "Flow bandwidth mlwdf for lte downlink transmission,".

[6] Z. Wei, B. Zhao, J. Su, and X. Lu, "Dynamic edge computation offloading for internet of things with energy harvesting: a learning method," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4436–4447, 2018.

[7] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computinga key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[8] P. Hu, H. Ning, T. Qiu, Y. Zhang, and X. Luo, "Fog computing based face identification and resolution scheme in internet of things," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 1910–1920, 2016.

[9] M. Qin, L. Chen, N. Zhao, Y. Chen, F. R. Yu, and G. Wei, "Power-constrained edge computing with maximum processing capacity for iot networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4330–4343, 2018.

[10] P. Mach and Z. Becvar, "Mobile edge computing: a survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

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

[12] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. Quek, "Offloading in mobile edge computing: task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

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

[14] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4177–4190, 2018.

[15] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: survey and implications," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 3, pp. 443–461, 2010.

[16] D. Mishra, S. De, S. Jana, S. Basagni, K. Chowdhury, and W. Heinzelman, "Smart rf energy harvesting communications: challenges and opportunities," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 70–78, 2015.

[17] D. Mishra, S. De, and D. Krishnaswamy, "Dilemma at rf energy harvesting relay: downlink energy relaying or uplink information transfer?" *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4939–4955, 2017.

[18] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1784–1797, 2017.

[19] H. Ke, J. Wang, H. Wang, and Y. Ge, "Joint optimization of data offloading and resource allocation with renewable energy aware for iot devices: a deep reinforcement learning approach," *IEEE Access*, vol. 7, Article ID 179349, 2019.

[20] M. Pasha and K. U. Rahman Khan, "Scalable and energy efficient task offloading schemes for vehicular cloud computing," *International journal of Computer Networks & Communications*, vol. 10, 2018.

[21] G. Cui, X. Li, L. Xu, and W. Wang, "Latency and energy optimization for mec enhanced sat-iot networks," *IEEE Access*, vol. 8, Article ID 55915, 2020.

[22] Y. Wei, F. R. Yu, M. Song, and Z. Han, "Joint optimization of caching, computing, and radio resources for fog-enabled iot using natural actor–critic deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2061–2073, 2018.

[23] X. Cheng, F. Lyu, W. Quan et al., "Space/aerial-assisted computing offloading for iot applications: a learning-based approach," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1117–1129, 2019.

[24] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, and A. Neal, "Mobile-edge computing introductory technical white paper. mob.-edge comput," *MEC) Ind. Initiative*, vol. 29, pp. 854–864, 2014.

[25] S. E. Mahmoodi, R. Uma, and K. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Transactions on Cloud Computing*, vol. 7, 2016.

[26] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[27] Y. Wu, L. P. Qian, H. Mao et al., "Secrecy-driven resource management for vehicular computation offloading networks," *IEEE Network*, vol. 32, no. 3, pp. 84–91, 2018.

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

[29] N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob, and M. Imran, "The role of edge computing in internet of things," *IEEE Communications Magazine*, vol. 56, no. 11, pp. 110–115, 2018.

[30] H. Liao, Z. Zhou, X. Zhao et al., "Learning-based context-aware resource allocation for edge-computing-empowered industrial iot," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4260–4277, 2019.

[31] L. Cui, C. Xu, S. Yang et al., "Joint optimization of energy consumption and latency in mobile edge computing for internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4791–4803, 2018.

[32] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for energy-efficient mobile edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4188–4200, 2018.

[33] Z. Zhou, H. Liao, X. Zhao, B. Ai, and M. Guizani, "Reliable task offloading for vehicular fog computing under information asymmetry and information uncertainty," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8322–8335, 2019.

[34] S. Li, N. Zhang, S. Lin et al., "Joint admission control and resource allocation in edge computing for internet of things," *IEEE Network*, vol. 32, no. 1, pp. 72–79, 2018.

[35] Y. Wang, J. Yang, X. Guo, and Z. Qu, "A game-theoretic approach to computation offloading in satellite edge computing," *IEEE Access*, vol. 8, Article ID 12510, 2019.

[36] X. Liu, A. Liu, T. Wang et al., "Adaptive data and verified message disjoint security routing for gathering big data in energy harvesting networks," *Journal of Parallel and Distributed Computing*, vol. 135, pp. 140–155, 2020.

[37] S. Bi, C. K. Ho, and R. Zhang, "Wireless powered communication: opportunities and challenges," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 117–125, 2015.

[38] F. Zhou, Y. Wu, H. Sun, and Z. Chu, "Uav-enabled mobile edge computing: offloading optimization and trajectory design," in *Proceedings of the 2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, Kansas City, MO, USA, May 2018.

[39] X. Liu, M. Zhao, A. Liu, and K. K. L. Wong, "Adjusting forwarder nodes and duty cycle using packet aggregation routing for body sensor networks," *Information Fusion*, vol. 53, pp. 183–195, 2020.

[40] Y. Sun, X. Guo, J. Song et al., "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3061–3074, 2019.

[41] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 3, pp. 361–373, 2017.

[42] L. Quan, Z. Wang, and F. Ren, "A novel two-layered reinforcement learning for task offloading with tradeoff between physical machine utilization rate and delay," *Future Internet*, vol. 10, no. 7, p. 60, 2018.

[43] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, "Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 64–69, 2019.

[44] Z. Ning, P. Dong, X. Wang, J. J. P. C. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 6, pp. 1–24, 2019.

[45] M. Qin, N. Cheng, Z. Jing et al., "Service-oriented energy-latency tradeoff for iot task partial offloading in mec-enhanced multi-rat networks," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1896–1907, 2020.

[46] X. Xu, D. Li, Z. Dai, S. Li, and X. Chen, "A heuristic offloading method for deep learning edge services in 5g networks," *IEEE Access*, vol. 7, Article ID 67734, 2019.

[47] J. Hu, Y. Li, G. Zhao, B. Xu, Y. Ni, and H. Zhao, "Deep reinforcement learning for task offloading in edge computing assisted power iot," *IEEE Access*, vol. 9, Article ID 93892, 2021.

[48] A. M. Seid, G. O. Boateng, S. Anokye, T. Kwantwi, G. Sun, and G. Liu, "Collaborative Computation Offloading and Resource Allocation in Multi-Uav Assisted Iot Networks: A Deep Reinforcement Learning Approach," *IEEE Internet of Things Journal*, vol. 8, 2021.

[49] Y. Wei, F. R. Yu, M. Song, and Z. Han, "User scheduling and resource allocation in hetnets with hybrid energy supply: an actor-critic reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 17, no. 1, pp. 680–692, 2017.

[50] V. Mnih, A. P. Badia, M. Mirza et al., "Asynchronous methods for deep reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, pp. 1928–1937, PMLR, New York, NY, USA, 2016.

[51] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[52] A. Tatar, M. D. De Amorim, S. Fdida, and P. Antoniadis, "A survey on predicting the popularity of web content," *Journal of Internet Services and Applications*, vol. 5, no. 1, pp. 1–20, 2014.

[53] S.-I. Amari, "Natural gradient works efficiently in learning," *Neural Computation*, vol. 10, no. 2, pp. 251–276, 1998.

[54] A. B. Labao, M. A. M. Martija, and P. C. Naval, "A3c-gs: adaptive moment gradient sharing with locks for asynchronous actor-critic agents," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, 2020.