electronics

MDPI

Article

Distributed Multi-Agent Approach for Achieving Energy Efficiency and Computational Offloading in MECNs Using Asynchronous Advantage Actor-Critic

Israr Khan [1], Salman Raza [2], Razaullah Khan [3], Waheed ur Rehman [4], G. M. Shafiqur Rahman [5] and Xiaofeng Tao [1,*]

[1] National Engineering Research Center for Mobile Network Technologies, Beijing University of Posts and Telecommunications, Beijing 100876, China; israr@bupt.edu.cn
[2] Department of Computer Science, National Textile University, Faisalabad 37610, Pakistan; salmanraza@ntu.edu.pk
[3] Department of Computer Science, University of Engineering and Technology, Mardan 23200, Pakistan; razaullah@uetmardan.edu.pk
[4] Department of Computer Science, University of Peshawar, Peshawar 25120, Pakistan; wahrehman@uop.edu.pk
[5] Key Laboratory of Universal Wireless Communications (Ministry of Education), Beijing University of Posts and Telecommunications, Beijing 100876, China; shafiq.it@hotmail.com
* Correspondence: taoxf@bupt.edu.cn

Abstract: Mobile edge computing networks (MECNs) based on hierarchical cloud computing have the ability to provide abundant resources to support the next-generation internet of things (IoT) network, which relies on artificial intelligence (AI). To address the instantaneous service and computation demands of IoT entities, AI-based solutions, particularly the deep reinforcement learning (DRL) strategy, have been intensively studied in both the academic and industrial fields. However, there are still many open challenges, namely, the lengthening convergence phenomena of the agent, network dynamics, resource diversity, and mode selection, which need to be tackled. A mixed integer non-linear fractional programming (MINLFP) problem is formulated to maximize computing and radio resources while maintaining quality of service (QoS) for every user's equipment. We adopt the advanced asynchronous advantage actor-critic (A3C) approach to take full advantage of distributed multi-agent-based solutions for achieving energy efficiency in MECNs. The proposed approach, which employs A3C for computing offloading and resource allocation, is shown through numerical results to significantly reduce energy consumption and improve energy efficiency. This method's effectiveness is further shown by comparing it to other benchmarks.

Keywords: deep reinforcement learning; advanced asynchronous advantage actor-critic (A3C); multi-agent system; mobile edge computing; cloud computing; computational offloading; energy efficiency

Citation: Khan, I.; Raza, S.; Khan, R.; Rehman, W.u.; Rahman, G.M.S.; Tao, X. Distributed Multi-Agent Approach for Achieving Energy Efficiency and Computational Offloading in MECNs Using Asynchronous Advantage Actor-Critic. *Electronics* **2023**, *12*, 4605. https://doi.org/10.3390/electronics12224605

Academic Editors: Bahman Javadi and Giovanni Crupi

Received: 9 July 2023
Revised: 5 September 2023
Accepted: 7 November 2023
Published: 10 November 2023

1. Introduction

The utilization of computation-intensive applications, such as augmented reality (AR), virtual reality (VR), gaming on the internet, smart transportation, industrial and residential automation, developed social networking, and facial recognition, has experienced a surge in popularity due to the widespread adoption of the internet of things (IoT) [1,2]. It is anticipated that this pattern will persist, given the predicted increase in the quantity of internet of things (IoT) devices, which is estimated to reach 24 billion by the year 2020 [3], and the projected surpassing of 30.6 exabytes per month in worldwide mobile network traffic. These numerous IoT entities, such as mobile users and sensors, will generate an unparalleled amount of data that need processing and analysis for the purpose of delivering the service. Undoubtedly, this massive amount of data represents enormous business opportunities. However, to process these unprecedented amounts of data and

to satisfy the computation-intensive applications, the system will consume a tremendous amount of computational energy resources and delays will be generated [1,3]. Therefore, to tackle computationally costly applications and minimize energy, mobile edge computing (MEC) has been introduced as a groundbreaking architecture embedded with cache and processing capacity in the vicinity of the user equipment (UE) [4–6].

Edge computing is considered a compliment to cloud computing technologies. It provides significant storage, computing, communication, and management capabilities at the edge. Multi-access edge computing has a remarkable capability to perform collaborative radio signal processing (CRSP) and collaborative radio resource management (CRRM) in real time for user equipment (UE) that is located nearby. On the other hand, C-RAN utilizes centralized signal processing and resource allocation to meet user demand efficiently [5]. Therefore, the MEC network concept has great potential in meeting diverse demands, especially in artificial intelligence (AI)-based future wireless networks [7]. Offloading efficient computations is a promising strategy to reduce network energy consumption and latency. In the context of computation offloading demand of IoT devices in MEC networks, edge computing can play a significant role by executing the computing tasks at the edge instead of sending the data to the remotely located cloud computing tier [8]. However, the computation offloading becomes immensely challenging due to the limitation of embedded computing resources of the edge access point (E-AP). Moreover, the computation offloading problem becomes intractable due to its stochastic nature. It is infeasible to tackle this problem with the old-fashioned algorithm, which takes a long time, unavoidably generates a delay, and consumes more energy [9].

In the past few years, the DRL (deep reinforcement learning) algorithm, a form of single-agent artificial intelligence, has garnered significant interest from the academic and industrial sectors. This is primarily due to its exceptional ability to control and solve highly complex problems [10,11]. In addition, the DRL method has been widely used for intelligent decisions on computational offloading and allocating resources in IoT networks. However, next-generation IoT networks are more decentralized, and the controller can make the decision in a decentralized manner to enhance the throughput while minimizing energy in these future wireless communication networks [12,13].

In this article, we investigate a computational offloading and resource allocation technique based on the asynchronous advantage actor-critic (A3C) algorithm. Each E-AP decides the suitable location for offloading the computational tasks in a distributed manner without the knowledge of global information of the network other than the surrounding E-APs. The A3C in DRL is a state-of-the-art technique that was developed based on the idea of multiple agents learning the task in a distributed manner and then being able to update the global policy asynchronously for performing the best action [11,14]. In A3C, the agent learns the stochastic policy rather than the deterministic policy (Q-Table), which makes it more robust than other asynchronous algorithms. Owing to the advantages of the MECNs and motivated by the parallel computing notion of A3C, in this article, we examine a multi-agent-based multi-layer computational offloading and resource allocation strategy for attaining energy efficiency in mobile edge computing networks (MECNs).

1.1. Related Work

Recently, researchers have focused on reducing delay and energy consumption through computational offloading and resource allocation techniques. In [11], the authors specifically examined how fog and cloud computing can work together to minimize service delay for IoT devices. To accomplish this, they optimized computational offloading, computing resource allocation, radio bandwidth allocation, and transmit power assignment, in a cloud/fog computing system. The objective was to reduce the weighted total of delay and energy consumption, and the task was described as a mixed-integer non-linear programming problem. The researchers used a sub-optimal approach with minimal complexity to address this problem, as described in [11,15]. To accelerate the performance of offloading, the learning-based offloading strategy goes one step ahead. A reinforcement learning-based

computing offloading strategy was investigated in [1,2], and [16] achieved a significant performance gain in IoT applications. In [17], the author analyzed a multi-user, multi-edge node computational offloading issue constrained by the availability of computing resources. Subsequently, a model-free reinforcement learning offloading method was applied to the time-varying channel information and CPU cycles to learn the long-term offloading approaches and enhance their utility. Due to the dimensionality curse and the limitations of Q-learning, the DRL method has been increasingly applied to huge state-space-based complex control issues.

Deep reinforcement learning (DRL) is investigated in [18] as a potential solution for tackling network dynamics, resource diversity, and the integration of managing resources with mode selection in mobile edge computing networks (MECNs). In [19], the authors presented a joint optimization challenge, where they addressed mode selection, channel assignment, power allocation, and discrete phase shift selection to optimize the average sum data rate of device-to-device pairs. In [12], the authors explore using a DRL approach to managing computation offloading and multi-user scheduling in internet of things (IoT) edge computing systems. The aim of this research was to reduce the overall delay and power consumption, taking into account the unpredictable arrival of data traffic, by optimizing the weighted sum of these two factors over an extended period of time. Similarly, in [20] employs an actor-critic DRL strategy to handle content caching, compute offloading, and radio resource allocation difficulties in edge-enabled IoT systems. They offer a combined optimization method for the edge-enabled IoT network to decrease energy usage and end-to-end delay. Moreover, in [21], the authors proposed a deep Q-network-based power allocation method for NOMA-enabled network devices to improve energy efficiency and reduce power consumption. They examined wireless network delays and dynamic energy-efficient resource allocation, optimizing the system's energy efficiency while meeting delay constraints.

The authors proposed a strategy in [11,22] for offloading computation tasks in mobile edge computing (MEC) networks by utilizing a double deep Q-network-based approach that considers the time-varying network dynamics. Their approach describes the computational offloading policy as a Markov chain of decisions with the objective of maximizing long-term utility. Another approach, i.e., deep reinforcement learning-based online offloading (DROO), is studied in [23] for precise offloading decisions and wireless resource allocations under the time-varying conditions of a wireless channel. DROO uses a deep neural network as a scalable method for advanced learning. Several studies have explored deep reinforcement learning for intelligent resource allocation and management, such as in [23], where a DRL-based communication link scheduling algorithm is explored for a cache-enabled opportunistic interference alignment wireless network. A DNN with multiple convolutional layers is utilized to represent the state, and features are extracted from a high-dimensional input containing CSI and cache states.

The challenge lies in finding the optimal offloading and resource allocation decisions, often formulated as a mixed-integer non-linear programming (MINLP) problem, which is NP-hard, in general [24]. Different approaches have been proposed to solve this problem, such as differentiable optimization techniques, metaheuristic algorithms, and DRL methods. Differentiable optimization techniques use gradient-based methods to find the optimal solution. However, they require the objective function and constraints to be differentiable, which may not be the case for some MEC scenarios [25]. Metaheuristic algorithms are stochastic optimization methods that explore the search space using various operators, such as mutation, crossover, and selection. They can handle complex and non-linear problems but may suffer from slow convergence and high computational complexity [26,27].

Most existing works mainly focus on dual-layer computation offloading. However, the demand for computation offloading is rapidly increasing in the connected IoT domain. Therefore, this paper evaluated the multi-layer computation offloading scheme in MECNs to fill the gap. Furthermore, to suppress the energy and delay, a state-of-the-art DRL-A3C multi-agent-based learning approach was exploited in this work. The A3C algorithm

leverages a collaborative setup involving a global network and multiple agents, making it an advanced tool for real-time learning and complex problem-solving. Particularly in scenarios involving the non-convex MINLFP problem in dynamic and dense wireless environments, the A3C algorithm demonstrates its suitability and practicality [28].

1.2. Contribution and Organization

The following is a summary of the principal contributions of this study:

- MECNs are currently exploring a problem related to energy optimization using DRL, which involves selecting the appropriate mode and allocating radio resources. This optimization problem considers various constraints, such as limited computing and radio resources, and quality-of-service requirements for individual user equipment (UE) in a constantly changing wireless environment;
- In order to minimize energy consumption in MECNs, a joint optimization method for mode selection, radio resource allocation, and distributed computing resource allocation is proposed. We modeled the mode selection and resource allocation problem under the Markov decision process (MDP) problem. Due to a large amount of system states and activities in our problem, we used the DRL approach, which employs a DNN as a function approximator to predict value functions. We implemented the fixed target network and experience replay technique to ensure a stable training process. In order to allocate the optimal number of computing resources, we utilized the greedy algorithm;
- Extensive simulation results demonstrate that the proposed DRL-based offloading and radio resource allocation mechanism can achieve a stable state very quickly and performs well in terms of energy efficiency in MECNs, while computing resource allocation is handled separately with a greedy algorithm. The proposed approach demonstrates better performance when compared to Q-learning, fixed and random approaches, and partial resource allocation schemes.

The article is organized as follows. The system model is described in Section 2, while Section 3 presents the analytical formulation of the energy optimization problem. Section 4 offers the DRL-based computing offloading policy. The simulation results are presented in Section 5, which validates the performance improvement of the proposed DRL-based computing offloading strategy. Finally, the paper is concluded in Section 6.

2. System Model

The suggested system architecture for resource allocation and compute offloading in uplink E-APs is shown in Figure 1. The network comprises a collection of E-APs $\mathcal{L} = \{1, 2, \ldots, L\}$ and the number pieces of UE $\mathcal{K} = \{1, 2, \ldots, K\}$. We assume that each E-AP has the storage capacity S_l, which can store the maximum amount of data, including libraries and databases [14]. All the notations used in the system model are listed in Table 1.

We consider there to be three shared environments, each encompassing a maximum of three E-APs. The E-AP is allocated with the best channel capacity and maximum resources and performs as an agent of its environment; the other two E-APs work as assistive E-APs. In the proposed model, three scenarios are considered based on the user's demand. Shared environment 1 illustrates that if the assistive E-APs are capable of satisfying the users' demand within the tolerable delay δ^{max}, UE will be served by the assistive E-APs where the agent E-AP is responsible for allocating the optimal resources and selecting the suitable E-AP based on user demand. According to shared environment 2, all the E-APs participate in executing the computation tasks where the agent distributes the UE among them. In shared environment 3, the agent E-AP is preferred to accomplish the computation since it has superior computing resources and channel capacity than the assistive E-AP. The proposed model considers two types of IoT devices. UE that has processing and cache capacities is considered to be smart user equipment (SUE), i.e., smartphones or laptops.

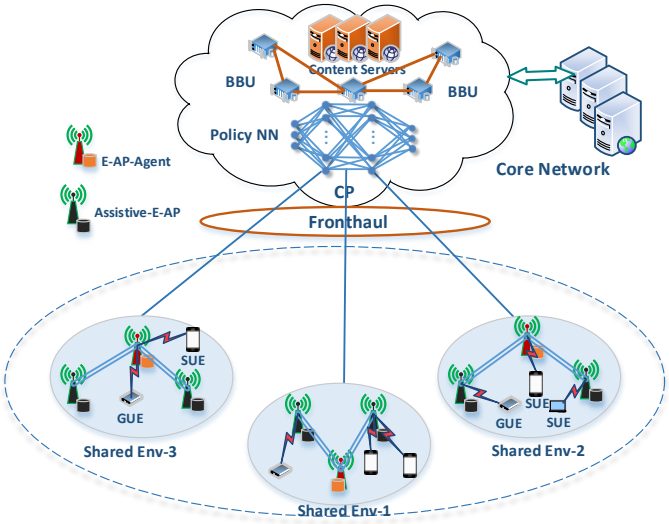

Figure 1. MECNs with deep reinforcement learning (DRL) for computation offloading.

Table 1. List of Notations.

Symbol	Definition
A3C	Advanced Asynchronous Advantage Actor-Critic
AI	Artificial Intelligence
DRL	Deep Reinforcement Learning
MECNs	Mobile Edge Computing Networks
CP	Cloud Processor
CRRM	Collaborative Radio Resource Management
CRSP	Collaborative Radio Signal Processing
E-AP	Edge Access Point
SUE	Smart User Equipment
CSI	Channel State Information
GUE	General User Equipment
s_k	transmitted signal vector
$p_{k,l}$	the transmitting power from user k to E-AP l
$h_{k,l}^H$	the channel gain between user k to E-AP l
s_k^m	available mode for user k
γ_{min}	the lower bound of QoS for user k
R_k	maximum achievable transmission rate
t_k^{max}	tolerable maximum latency for the task completion
M_k	task of user k
$R_{k,l}$	transmission rate between user k to E-AP l
$R_{k,j}$	transmission rate between user k to RRH j
$f_{l,k}$	the allocated computing resources of E-AP l to user k
ω_l	energy consumption for per CPU cycle at E-AP l
s	state in the environment
a	the agent action
ξ	the discount factor
r	reward obtained from appropriate action
$(s,a)_t$	action-state pair for every time t cycle
$Q^*(s_t,a_t)$	optimal Q-function at time t
$r(s,a)$	the reward obtained from the current action state pair
α	the learning rate of Q-learning
$L_i(\omega_i)$	The loss function in which ω represents the parameter of the neural network
E	Expected outcome

On the other hand, UE without processing and cache capacities is regarded as general user equipment (GUE), which includes sensors and industrial monitoring devices. In the context of computation offloading, UE sends the offloading request to the closest E-AP.

After receiving a service request from the SUE, the agent rigorously analyzes the task to take the best possible action. If the task is suitable for execution by the SUE with its embedded local resources, it will be responsible for executing it. In contrast, if the task is beyond the SUE's processing capacity, the task will be completed at the edge through an individual or group of E-APs. Despite the joint processing of E-APs, if the request cannot be finished, then the cloud computing zone will be accountable for tackling it by default. On the other hand, the GUE directly offloads its computation tasks in the edge or cloud computing zone based on the controller's decision. We assume that each piece of UE and E-AP has a single antenna. We consider that the global actor-critic network is located in the cloud tier, and each internal agent (worker) is located at the edge with the same parameter of the global network. The proposed system architecture rigorously considers the path loss, shadowing, and fast-fading model.

2.1. Task Model

We assume that each piece of UE k has the computation task at time slot t as $\phi_k^t = \{C_k^t, M_k^t, t_k^{max}\}$, which can be executed on the smart UE (SUE) by employing local computing resources or offloading either to the E-AP or cloud computing tier to accomplish the computation. C_k^t denotes the required computing resources (total number of CPU cycles) to accomplish the computations task ϕ_k^t, M_k^t is the size of input data for computing ϕ_k^t including the input parameters and program codes, and t_k^{max} represents the tolerable maximum latency for the task completion. Furthermore, we assume that the task of each UE k will be intact and impossible to break into partitions. As a result, each request will be executed on only one layer. Hereafter, the task execution decision can be described as $S_k^m \in \{0, 1\}$ where $S_k^m = 1$ shows how the task will either be carried out locally by the SUE, at the edge using the E-AP, or on the cloud computing tier.

2.2. Cache Model

The primary objective of this work was to serve the rigorous computation demand in the vicinity of the UE by caching the computation-intensive services. The service is considered as the abstraction of applications, namely, video streaming, social gaming, navigation, and AR and VR, hosted by the E-APs or cloud and requested by the UE. To run a specific service, it is necessary to cache the related data, which includes required libraries and databases, on the service provider (smart devices, E-APs, or cloud computing). We consider each E-AP l to have the storage capacity S_l, which can be utilized to store the maximum number of services [14]. We will assume that a set is denoted as Q, which encompasses various computing services. These services are indexed using the set $\mathcal{Q} = 1, 2, \ldots, Q$. At the time slot t, the controller E-AP l, which decides whether the service $q \in Q$ should cache or not, can be represented as the binary variable $C_{q,l}^t \in \{0, 1\}$. However, insufficient cache storage capability limits the number of services at each E-AP l, which can be described as

$$\sum_{q \in Q} C_{q,l}^t s_q \leq S_l, \forall l, \forall t, \tag{1}$$

where s_q denotes the occupied storage space in E-AP l for caching the service q and the maximum storage capacity of E-APl is represented by S_l. Since the services considered are stored distributively among the E-APs in the shared environment, we strictly follow the non-duplication policy as follows:

$$\sum_{l \in n} \sum_{q \in Q} C_{q,l}^t s_q \leq 1, \forall q, \forall l, \tag{2}$$

where $C^t_{q,l} s_q \leq 1$ shows that only one service q can be cached in one E-AP l in the E-APs group n.

$$c_{f,l} = \begin{cases} 1 & \text{if the file } f \text{ cached in E-AP } l \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

2.3. Communication Model

Let us assume the SUE k cannot accomplish the computation task ϕ_k with the limited embedded resources, or the GUE needs to offload the tasks for computation. In this context, the UE must offload their tasks to either the E-AP or CP tier. Subsequently, the UE denoted as k transmits its service request to the closest E-AP, identified as l. The signal originating from the UE k when received at the E-AP l can be mathematically represented as follows:

$$y_{k,l} = \boldsymbol{h}^H_{k,l} p_{k,l} s_k + \sum_{i \neq k, i \in K} \boldsymbol{h}^H_{k,l} p_{i,l} s_i + n_l, \forall k, l, \tag{4}$$

Here, the $\boldsymbol{h}^H_{k,l}$ represents the channel gain between UE k and E-AP l, where the path loss, shadowing, and fast fading are considered. The transmission power for the uplink of the users i and k can be represented by $p_{i,l}$ and $p_{k,l}$, respectively. The received Gaussian noise at the E-AP l is represented by n_l, and it follows a distribution of $(0, \sigma^2_i)$. Moreover, the transmitted signal vector is represented by s_k, which essentially refers to the message from the user k [29].

The proposed model for computation offloading and resource allocation, which utilizes the A3C approach, dramatically relies on the signal-to-interference-plus-noise ratio (SINR). This metric plays a vital role in assessing the channel capacity and ensuring the quality of services (QoS). Specifically, the SINR for the E-AP l originating from the user k is calculated using the following formula:

$$SINR_{k,l} = \frac{|\boldsymbol{h}^H_{k,l}|^2 p_{k,l}}{\sum_{i \neq k} |\boldsymbol{h}^H_{i,l}|^2 p_{i,l} + \sigma^2_l}, \forall k, l, \tag{5}$$

where $p_{k,l}$ stands for the transmission power employed between the user k and E-AP l. It is worth highlighting that the transmit power $p_{k,l}$ might not remain uniform across all users, as variations can occur based on factors such as channel conditions and the power constraints imposed by the E-APs. We define γ_{min} as the target SINR to ensure seamless communication in our model. Consequently, for choosing the channel edge, the threshold γ_{min} must be satisfied. The appropriate transmission rate also propels the system to guarantee the QoS and reduce the transmission or communication delay. Taking this into consideration and following the Shannon channel capacity theory, the attainable transmission rate in the edge can be characterized as:

$$R_{k,l} = W \log_2(1 + \gamma_{min}), \tag{6}$$

2.4. Computation Model

In the proposed system architecture, two sorts of delays are rigorously considered: computation and transmission delays. The processing of UE's tasks k in several tiers, such as the SUE, the E-AP, and the cloud computing zone, causes the computation delay. On the other hand, the fronthaul-based uploading of jobs M_k from the UE to the cloud or from the UE to the edge results in transmission latency. For downloading the computed data, the system also produces a delay. The following sub-sections offer a comprehensive breakdown of the delays incurred.

2.4.1. Local Execution

Based on the decision ($s^{local}_k = 1$) of the agent l, the SUE k performs the computation locally, where the computation time depends on the implanted resources of the SUE. As-

sume for the moment that f_k^{local} represents the SUE's computing power, which is evaluated regarding the count of CPU cycles executed per second [1,2]. $0 \leq \delta_k \leq t_k^{max}$ is the maximum allowable computation time. Therefore, the local computation delay can be expressed as follows:

$$D_k^{local} = \frac{C_k}{f_k^{local}}, \tag{7}$$

where D_k^{local} is the processing delay for accomplishing the computing tasks ϕ_k and C_k represents the total required CPU cycle.

$$E_k^{local} = \zeta_k C_k \tag{8}$$

The coefficient ζ_k represents the energy consumption per CPU cycle and depends on the UE's chip architecture. In this study, we set ζ_k equal to 10^{-27} times the square of the local frequency, based on the research presented in [30].

2.4.2. Edge Execution

We assume that, due to the limited resources, the SUE k is not able to accomplish the computing task within the time frame of t_k^{max}, whereas the GUE needs to offload the tasks for computation. In both cases, the agent l rigorously analyzes the task and takes the initiative to execute the task at the edge ($s_k^{edge} = 1$) aiming to achieve extremely low latency. Besides the mode selection, the agent l is also accountable for choosing the optimal E-AP based on the UE demand. In the case of offloading the task to the edge, the total delay specifically involves the computation delay and uplink transmission delay [1]. The cumulative latency experienced at the edge is expressed as follows:

$$D_k^{edge} = \frac{M_k}{R_{k,l}} + \frac{C_k}{f_{l,k}^{edge}}, \tag{9}$$

where $f_{l,k}^{edge}$ is the amount of computation resources that the E-AP l assigns to UE k and $R_{k,l}$ stands for the attainable transmission rate of UE k for accessing the E-AP l. The corresponding energy consumption due to transmission is expressed as:

$$E_k^{edge} = p_{k,l} \frac{M_k}{R_{k,l}} \tag{10}$$

2.4.3. Cloud Execution

Since the edge node operates within the constraints of limited resources within the E-RAN [5,7]. In some cases, the computational requirements of UE k goes beyond the computation resources ($f_{l,k}^{edge}$) of E-AP l. Therefore, to tackle the computation demand, the agent l selects the cloud tier ($s_k^{cloud} = 1$), and the UE k offloads its task through C-RAN mode. Generally, the delay associated with offloading to the cloud tier encompasses both an uploading delay and a processing delay. This can be formulated as follows:

$$D_k^{cloud} = \frac{M_k}{\sum_{j \in J} R_{k,j}} + \frac{M_k}{\sum_{j \in J} R_{j,CP}} + \frac{C_k}{f_{CP,k}^{cloud}}, \tag{11}$$

where $R_{k,j}$ signifies the achievable transmission rate from the UE k to RRH j and $R_{j,CP}$ represents the transmission rate from RRH j to the cloud processor CP. $f_{CP,k}^{cloud}$ illustrates that the CP allocates f amount of computing resources to the UE k. It is important to note that in our work, the volume of data uploaded is notably larger than that of processed data. Since the RRH is near the UE, the downloading delay from edge nodes to the UE is negligible. However, given that the cloud server is positioned at a considerable distance and is connected through fiber with the core networks [1], the downloading delay from the

CP to RRH is significantly non-negligible in this E-RAN system. The total offloading delay for the cloud tier can be expressed as:

$$D_k^{cloud} = \frac{M_k}{\sum_{j\in J} R_{k,j}} + \frac{M_k}{\sum_{j\in J} R_{j,CP}} + \frac{C_k}{f_{CP,k}^{cloud}} + \frac{M_k^{processed}}{\sum_{j\in J} R_{CP,j}}, \tag{12}$$

where $M_k^{processed}$ is the output data after processing at the cloud servers. The corresponding energy consumption due to transmission is expressed as:

$$E_k^{Cloud} = p_{k,j}\left(\frac{M_k}{\sum_{j\in J} R_{k,j}} + \frac{M_k}{\sum_{j\in J} R_{j,CP}}\right) \tag{13}$$

It is worth mentioning that each piece of UE k can choose a single mode to accomplish their computation demand. In this condition, the computation will be performed at the local computation tier (SUE), cloud, or edge computing tier. The overall energy consumption conditions can be represented as follows:

$$E_k^{service} = s_k^{local} E_k^{local} + s_k^{edge} E_k^{edge} + s_k^{cloud} E_k^{cloud}, \tag{14}$$

where $s_k^{local} + s_k^{edge} + s_k^{cloud} = 1$ and $s_k^{local}, s_k^{edge}, s_k^{cloud} \in \{0,1\}$; if $s_k^{local} = 1$ the task will be executed by the SUE by itself. If $s_k^{edge} = 1$, the computation process will be accomplished at the edge tier. On the other hand, if $s_k^{cloud} = 1$, the computation tasks will be executed at the cloud computing zone. Furthermore, condition $s_k^{local} + s_k^{edge} + s_k^{cloud} = 1$ strictly denotes that the user k can opt for only one mode based on the computation demand.

3. Problem Statement

In this section, we formulate the optimization problem for computation offloading and resource allocation in mobile edge computing networks. The primary objective of this problem is to minimize the total energy consumption across the system while simultaneously achieving extremely low latency. This is achieved by optimizing the computation offloading strategy for each user s_k^m, determining the computation resource allocation at the edge $f_{l,k}$, and managing the radio resource allocation p_k while adhering to the maximum delay tolerance set for each task of the IoT device t_k^{max}. The problem statement is outlined as follows:

$$\begin{aligned}
&\min_{\{s_k^m, f_{l,k}, p_k\}} E_k^{service} \\
\text{s.t.}\quad &\text{C1}: s_k^m \in \{0,1\}, k \in \mathcal{K}, m \in \{local, edge, cloud\}, \\
&\text{C2}: s_k^{local} + s_k^{edge} + s_k^{cloud} \le 1, k \in \mathcal{K}, \\
&\text{C3}: \textstyle\sum_{k=1}^{K} s_k^{edge} f_{l,k} \le f_l^{max}, \forall k \in \mathcal{K}, \\
&\text{C4}: s_k^{edge} SINR_{k,l} \ge \gamma_{min}, \forall k \in \mathcal{K}, \\
&\text{C5}: s_k^{cloud} SINR_{k,j} \ge \gamma_{min}, \forall k \in \mathcal{K}, \\
&\text{C6}: \textstyle\sum_k s_k^{edge} p_{k,l} \le P_{max}, k \in \mathcal{K}, \\
&\text{C7}: \textstyle\sum_k s_k^{cloud} p_{k,j} \le P_{max}, k \in \mathcal{K}, \\
&\text{C8}: D_k^{service} \le t_k^{max}, k \in \mathcal{K}
\end{aligned} \tag{15}$$

where the constraint (15) implies that the user k is allowed to choose only one mode at a time. $C3$ denotes that the frequency of each E-AP f_l is constrained by maximum frequency f_l^{max}, which is the number of users that can be scheduled to each E-AP l. $C4$ ensures the QoS requirements for better system performance by satisfying γ_{min}, which is considered as a lower bound. The computing power consumption at the edge cannot go

beyond the allocated power P_{total}, which is reflected by *C*6. Similarly, the *C*7 shows the upper bound of uplink transmission to be P_{max} allocated power. The objective variables of the optimization problem shown in (15) consist of p_k, $f_{l,k}$, and s_k^m, which are power allocation, computing resource allocation, and mode selection, respectively. The traditional approaches are incapable of solving the non-convex mixed-integer non-linear fractional programming problem (MINLFP) as presented in (15). It is especially challenging regarding the dynamic and dense environment of wireless networks. Due to the non-trivial nature of the problem due to the high demand for energy minimization, a distributed learning-based A3C scheme is advocated to resolve (15). A3C is a cutting-edge scheme based on distributed learning with multiple independent agents with their weights. They interact in a complex environment with multiple copies of the environment in parallel toward achieving a common goal [31].

4. Distributed Computation Offloading and Resource Allocation

Due to the continuous performance of single-agent-based centralized deep reinforcement learning, a large community has been focusing on extending the single-agent-based solution to a multi-agent-based fully distributed learning framework to make the system more robust [32]. Consequently, Mnih et al. proposed an A3C algorithm, which is simpler, faster, more robust, and able to achieve much better scores under the light of DRL tasks [33]. Unlike the single agent DQN, A3C distributes the learning tasks to several agents, where each agent asynchronously updates a deep neural network (DNN) based on its independent learning experience. The most significant reason behind the extraordinary performance of A3C is the incorporation of multiple agents that experience diverse situations and share their experience with a global network (global model) connected to all the agents. The global model updates using a gradient, which is collected from each agent, and then shares its most recent global weights with all agents [34]. Furthermore, the A3C exploits the standard actor-critic learning asynchronously. In the proposed study, we developed the computation offloading and resource allocation scheme where each agent E-AP distributively selects precise offloading and allocates an optimal amount of computation and radio resources.

4.1. Computation Offloading is Modeled as a DRL Problem

In DRL, an agent engages with the environment and performs actions to optimize the reward [35]. To achieve this, the state, action space, and reward function are defined as follows:

4.1.1. State Space

The system state constitutes the factors such as the available offloading mode user k, the maximum transmission rate, and the available distributed computing resources at an edge. Therefore, the system state is represented as $s_t = \{s_k^m, f_1^k, f_2^k, \ldots, f_L^k, R_k\}$, where s_k^m represents the cloud and edge computing mode for the user k. The maximum achievable transmission rate, R_k, is dominated by the E-AP and cloud modes' power resources. Distributed computing resources $\sum_{l \in L} f_{l,k}$ can be aggregated to meet the computation demand for the user k [36].

4.1.2. Action Space

Each agent performs a sequential action in the given environment based on the condition. The A3C is suitable for both the continuous and discrete state space and action space [33,37] . Furthermore, since each agent learns from its piece of the environment, we assume that the system will be faster.

Theoretically, the agent is capable of performing numerous actions. However, for performing a significant number of actions, the system requires enormous computation, which produces a tremendous amount of delay with high usage of energy, decreasing the system's performance. Therefore, to avoid the computation complexity, it was considered

that, based on the current state s_t, the agent performs the action a_t including the mode selection and power allocation for each time slot $t \in T$. The action space is represented as $a_t = \{s_k^m, p_k\}$. Based on the offloading decision, the controller allocates the power for the E-AP $p_{k,l}$. Furthermore, in each decision epoch t, the controller rigorously considers the computing resources $\sum_{l \in L} f_{l,k}$ at the edge, which are allocated through the greedy algorithm. In each action, the controller needs to satisfy the constraints.

4.1.3. Reward Function

The goal of a reward is to provide feedback to an RL model regarding the performance of a previous action. Therefore, it is essential to define the reward appropriately for the proper learning process. The reward function defined precisely and effectively helps to find the best action policy [35]. Efficient computation offloading can help address the energy optimization problem, which is closely related to the proposed reward function. The immediate reward is considered as the negative sum cost of energy. Precise mode selection, optimal computing resource allocation, and power allocation are crucial factors that significantly influence the immediate reward r_t. As a result, the immediate reward can be expressed as:

$$r_t = -\left(E_k^{service}\right) \tag{16}$$

We consider the long-term accumulative reward as energy minimizing at a more extended period of time T. The controller performs the best actions using exploration and exploiting all possibilities to maximize future accumulative rewards. Assuming a discount factor of $\xi \in [0, 1]$, the long-term cumulative reward R_t can be defined as $R_t = \sum_{t=0}^{T} \xi^t r_t$. The ξ is the indicator of future reward, which will influence the future reward based on the current mode selection; when the ξ value is lower, it will provide the priority for accumulating the immediate reward. Since our ultimate object is to minimize the energy, the reward is correlated with the negative value of the energy. Afterward, we endeavor to maximize the reward in order to minimize the energy.

4.2. Distributed Computing Resource Allocation

To overcome the resource limitation of E-APs, we analyzed the distributed computing scheme, where the available E-APs participate in executing the computing tasks and tackle the immense computation demand. In the dynamic computation offloading process, we exploited the greedy algorithm to allocate computing resources at the edge to avoid the rigorous time consumption of DRL training. The distributed computing resource allocation problem is considered a binary problem where the number of computing resources can overcome this challenge. Therefore, the simplified form of the original computing resource allocation problem can be stated as

$$\begin{aligned} \min_{\{f_{l,k}\}} \quad & p_{l,k} \frac{C_k}{\sum_{l=1}^{L} f_{l,k}}, \\ \text{s.t.} \quad & (15), \\ & E_{i,j} \leq \delta_E, \\ & E_k^{edge} \leq \varphi_E \end{aligned} \tag{17}$$

where the constraint (17) illustrates that the energy between the assistive E-AP and primary E-AP must be less than δ_E and the computation energy at the edge cannot go beyond the threshold delay of φ_E, as depicted by (17). In Algorithm 1, we further elaborate on our concept of maximizing the processing capacity at the edge and how to lessen the intervention of cloud computing in serving the computing demand. At steps 8 and 9, we calculate the requested computing task of users K and available computing resources at the edge, respectively. Hereafter, the computation energy is calculated according to (9), where the available computing resources are distributed to satisfy the computing demand. At steps 11 to 20, we allocate the computing resources and evaluate the energy at the edge

based on $E_{i,j} \leq \delta_E$ and $E_k^{edge} \leq \varphi_E$ [38]. We accept the result and consider the edge to be a suitable mode for offloading the task of the user k. Finally, the algorithm sends the edge status to the DQN for evaluation and future decisions.

Algorithm 1 A3C Algorithm for Distributed Computation

Input:
Set of users: $\mathcal{K} = \{1, 2, 3, \ldots, K\}$
Set of E-APs: $\mathcal{L} = \{1, 2, 3, \ldots, L\}$
Generated task for each user: M_k
Computing capacity of each E-AP:f_l
Tolerable computing energy is set to φ_E
Tolerable transmission energy from primary E-AP i to assistive E-AP j is set to δ_E;
Determine the total number of tasks requested by user K as $\sum_{k=1}^{K} M_k$;
Calculate the total available computing resources at the edge as $\sum_{l=1}^{L} f_l$
Output:
The selected offloading mode for each user's tasks that meet the specified energy and resource constraints
Iteration:
for *user* k: K **do**
    while $f_l \leq \sum_{l=1}^{L} f_l$ **do**
        if $E_{i,j} \leq \delta_E$ **then**
            Select the E-AP j as the suitable assistive E-AP for offloading the task of k
            Calculate computing energy at the edge according to (10)
            if $E_k^{edge} \leq \varphi_E$ **then**
                break
                Select edge is the suitable mode for offloading the task of k
            end if
        end if
        $f_l = f_l + 1$
    end while
end for
Return result

4.3. DRL-Based Offloading

Hence, we provide the offloading technique based on the DRL to improve computing performance, where the DRL functions as an agent, known as a deep Q-network, which combines the traditional RL and DNN to speed up the learning process [4]. For each action in DRL, we have a separate output. State representation is used as the only input to the neural network. DRL is renowned for its ability to reduce the complexity of a system by calculating the Q-values for all conceivable actions for a given state with a single forward pass [3,4]. In addition, better implementation is achieved by utilizing the replay memory and intrinsic generalization capabilities introduced by NN DRL to the minimum interaction with the complex environment. Q-learning in RL often results in an impractical situation. An approximator is used in DRL to estimate the action-value function $Q(s, a; \omega) \approx Q^*(s, a)$), which notably enhances the system's performance [4]. In Algorithm 2, the training procedure and offloading mechanism are displayed.

Algorithm 2 DRL Algorithm for Computing Offloading in E-RAN

Input:
set random weights ω for the Q-network $Q(s,a)$ initialization
build a weighted $\hat{\omega}$ target Q-network $\hat{Q}(s,a)$
Assign a capacity of N_D to the replay memory D
Determine the mini-batch size M for training network
Establish the maximal training session $Emax$
Output:
Trained Q-network ($Q(s,a)$)
Begin Iteration:
for each episode within E_{max} **do**
    Establishing the standard parameters of simulation for the computational offloading environment
    assign the starting state $s_0 = [s_k^{mode}, f_l^k, R_k, M_k]$
    for each *decision step* $t = 1 : T-1$ **do**
        Establish a random value xx between 0 and 1
        if $x \leq \varepsilon$ **then**
            Randomly select action a_t for mode selection and power allocation as $[s_k^m, p_k]$
            if $s_k^m == edge$ **then**
                Run algorithm (1) and allocate optimal computing resources to the user k
            else if $s_k^m == cloud$ **then**
                Run algorithm (1) and allocate optimal computing resources to the user k
            else
                Execute the task with local resources
            end if
        else:
            Take the best possible action a_t
            $a_t = \arg\max_{a_t \in A} Q(s_t, a_t; \omega)$ for selecting mode and allocating power as $[s_k^m, p_k]$
        end if
        Offload the task M_k of k-th user
        Perform action a_t and assess the system's energy
        Examine the reward r_t based on the formulated problem described in (16).
        Put the reward r_t along with s_t, s_{t+1}, and a_t as an interaction sample (s_t, a_t, r_t, s_{t+1}) in the replay memory D.
        From the replay memory D, randomly sample the mini-batch with the transitions (s_t, a_t, r_t, s_{t+1}) of size M.
        By using mini-batch descent gradient on $(r_t + \xi \max_{a' \in A} \hat{Q}(s_{t+1}, a'; \hat{\omega}^{)} - Q(s_{t+1}, a_t; \omega))^2$ with ω, to train the Q-network.
        Update the target Q-network with parameters ω to $\hat{\omega}$ periodically.
    end for
end for

The input for the deep neural network (DNN) comprises various parameters related to the current state of the user k, including available communication modes, radio and computing resources at the edge, and the size of the offloading work $[s_k^m, f_l^k, R_k, M_k]$. This information is depicted in both Figure 2 and Algorithm 2. The output formulates the $Q(s,a,\omega)$ with the weight of ω to achieve a more precise estimation based on all possible actions. To establish a balance between exploitation and investigation, the deployed IoT devices choose the offloading policy $a_t \in A$ based on the DNN result, which was received via a ε-greedy policy. Finally, the agent takes action to select the offloading mode s_k^m and assigns the adequate amount of power p_k, and offloads the task M_k. When the agent selects the edge mode s_k^{edge}, the system goes through Algorithm 1 and allocates an optimal number of computational resources to meet the computing requirements. If not, the request is processed using C-RAN mode at the cloud computing layer. The system transitions to a

new state s_{t+1} upon completion of an action a_t. In the meantime, the agent computes the reward r_t according to (16) for delay reduction based on the chosen modes. This transition $(st, at, rt, st+1)$ is stored as an experience in the replay memory D. The DQN will be updated at each iteration using a randomly picked batch of $M = 32$ elements. The DQN will undergo extensive training to achieve the target value, which will involve performing gradient descent and minimizing the loss. The following equation can be used to describe the loss function [39]:

$$L(\omega) = \mathbb{E}_{s,a,r,s'}\left[\left(r_t + \xi \max_{a' \in A} \hat{Q}(s_{t+1}, a'; \hat{\omega}) - Q(s_{t+1}, a_t; \omega)\right)^2\right] \quad (18)$$

The target network [39] is denoted by $(r_t + \xi \max_{a' \in A} \hat{Q}(s_{t+1}, a'; \hat{\omega})$ in the above equation. The agent will periodically update the network by adjusting the weights of the DQN to match those of the target DQN.

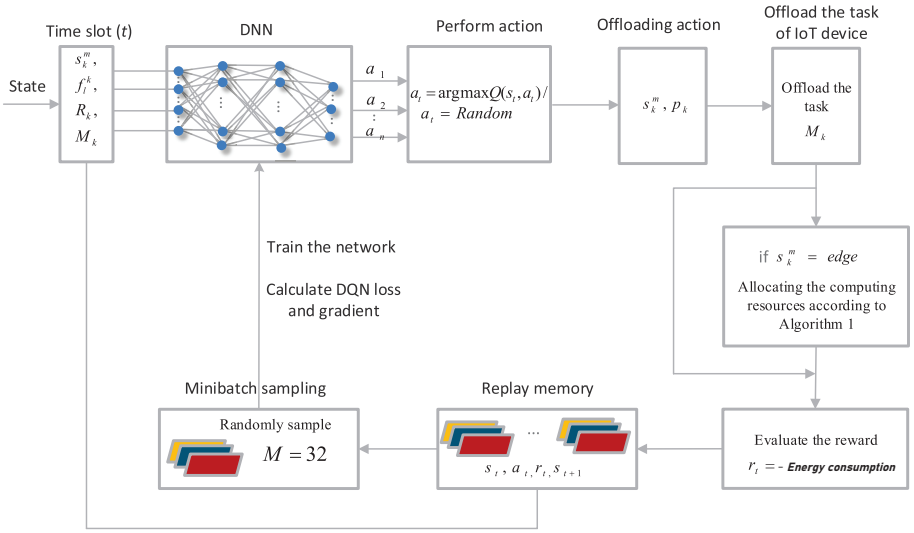

Figure 2. DRL-based learning process of individual agent.

5. Simulations Results and Discussion

This section aims to assess the effectiveness of the suggested DRL approach for simultaneous computation offloading along with resource allocation in IoT devices, as it was designed to minimize energy consumption in E-RAN. Furthermore, the greedy algorithm was also evaluated as a precursor for computing resource allocation through numerical upshots.

For the simulation platform, TensorFlow 1.11.0 and Python 3.6 were utilized, with an 8 CPU core i5 @ 1.6 GHz along with Intel UHD graphics 620. The system under evaluation assumes a maximum transmission power of 23dBm for each piece of user equipment (UE) and a deployment area of 400 × 400 m, and the system performance was assessed using 10 E-APs and 10 IoT devices. It was assumed that the path loss model is given by $128 + 37 * \log10(d)$ [40], where d is the distance between nodes. Additionally, the noise power was −174 dBm and the system bandwidth was set to 20 MHz. The QoS demand for both edge and cloud mode was regarded as min. Our computational model utilizes a fully connected deep neural network (DNN) architecture. The (DNN) consists of four discrete layers, namely, two hidden layers, an input layer, and a single output layer. This design allows the network to process and transform input data through these layers to generate the desired output. The first hidden layer consists of 64 neurons, while the second hidden layer has 32 neurons. The ReLU is used as the activation function in the hidden layers. All the necessary parameters of the simulation are listed in Table 2.

Table 2. Experimental parameters.

Number of E-APs	10
Number of users (K)	10
Power noise (σ^2)	-174 dBm
Maximum transmit power	23 dBm
Bandwidth (W) of channel	20 MHz
Pathloss model	$128 + 37 * \log_{10}$(distance) [40]
Experience replay buffer size N_D	2000
Learning rate α	0.01
ε-greedy	0.9
Reward decay ξ	0.9 [29]
Mini-batch size M	32

The proposed scheme was evaluated against the following baseline schemes:

1. Random scheme: the random algorithm involves making choices or decisions without considering any specific criteria;
2. Fixed scheme: The fixed algorithm follows pre-determined rules or fixed strategies for decision-making. It was designed based on pre-defined guidelines and does not adapt to changes in the environment or task requirements;
3. Q-learning scheme: The Q-learning allows dynamic cloud-edge selection. The system learns via interactions, updating the Q-table to estimate rewards for actions in different states;
4. DRL-based computation offloading and resource allocation scheme (DRL-CORA) [36]: The DRL-CORA scheme employs a DQN scheme that enables the system to choose between cloud and edge computing modes dynamically. Through iterative interactions with the environment, the system refines its neural network's Q-values to approximate anticipated rewards associated with diverse actions across varying states.

Figure 3 illustrates the correlation between the variation in offloading energy consumption and the number of tasks in different modes. The jointly offloading mode dominated the performance gain since the local, edge, and cloud modes execute the tasks simultaneously. It can be observed from Figure 3 that, when compared to C-RAN and edge mode, the jointly offloading mode minimized the energy consumption by approximately 39% and 42%, respectively. Moreover, the jointly offloading mode converged when the number of tasks was 90, with much less energy consumption than the other modes. Due to the controlled resources of E-RAN nodes, the energy was linearly increased at the edge. C-RAN mode also showed poor performance because of long-distance and constrained fronthaul.

Figure 4 illustrates the comparison between the average energy consumption of our suggested scheme and the benchmark scheme, as it relates to the varying number of computation tasks. However, it is evident from Figure 4 that, as the number of computation tasks increased, the energy consumption of all the schemes also increased. In addition, the proposed strategy takes into account the advantages of local, edge, and cloud computing in order to mitigate energy consumption. Additionally, it is worth noting that the proposed technique had the lowest energy use across all computational activities. Moreover, the proposed scheme started convergence when the number of tasks exceeded 90.

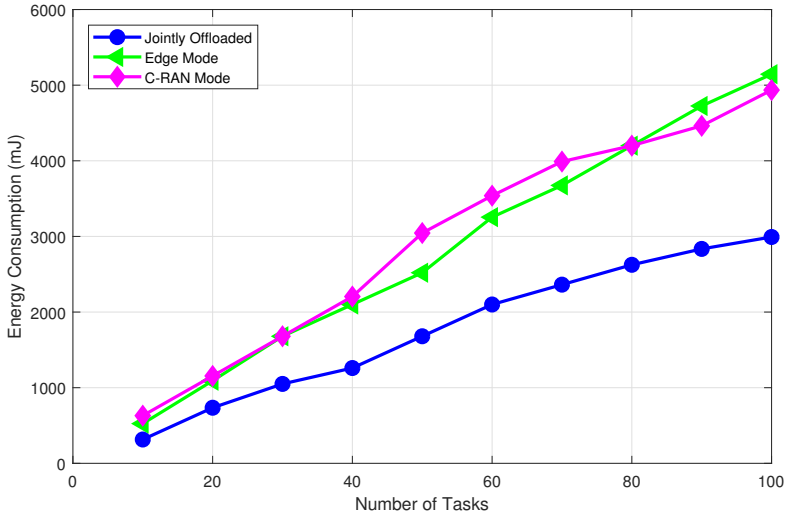

Figure 3. Evaluation of offloading energy consumption under different modes.

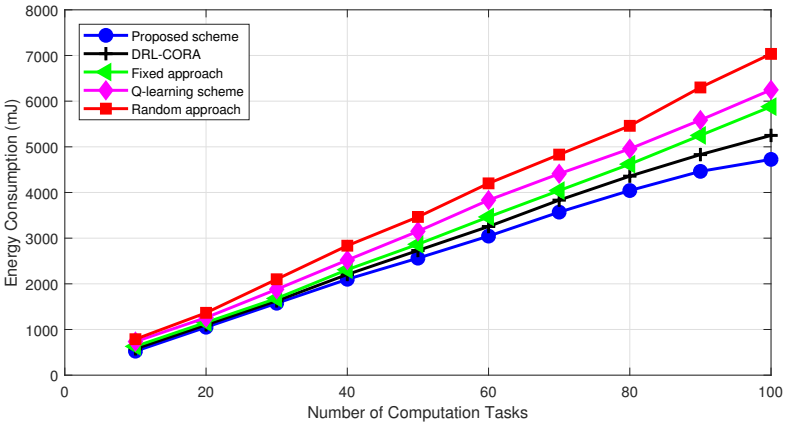

Figure 4. The evaluation of computing energy versus number of computational tasks.

Figure 5 depicts the performance of the number of computing resources with the offloading energy consumption. This analysis considers a scenario with 20 E-Aps with a processing capacity of 1.5 GHz each. The proposed approach achieved the lowest energy consumption from the schemes examined because of the DRL's remarkable controlling capabilities. Moreover, the DRL-CORA scheme exhibited superior performance compared to both the Q-learning and fixed approaches. In addition, the Q-learning and fixed approach saved nearly the same energy and showed almost the same trend in Figure 5. On the other hand, the random approach acquired more energy due to its random decision and exploitation behavior.

The energy consumption associated with tasks with varying completion time constraints in the application request is illustrated in Figure 6. As depicted in Figure 6, tasks with more lenient completion time constraints resulted in lower energy consumption. The graph indicates that, after a certain threshold, the time constraint results we extended for a similar offloading energy consumption for all methods. This phenomenon can be attributed to the less stringent task requirements and sufficient availability of resources, resulting in similar task assignments across all methods. Therefore, the energy consumption across different methods was highly similar. It is imperative to highlight that our proposed scheme consistently outperformed the other methods, particularly across the entirety of task completion time constraints. This superiority is particularly evident when the completion deadline surpassed 20 ms, confirming our algorithm's rapid convergence. This further underscores the robustness and effectiveness of our proposed approach in optimizing energy consumption.

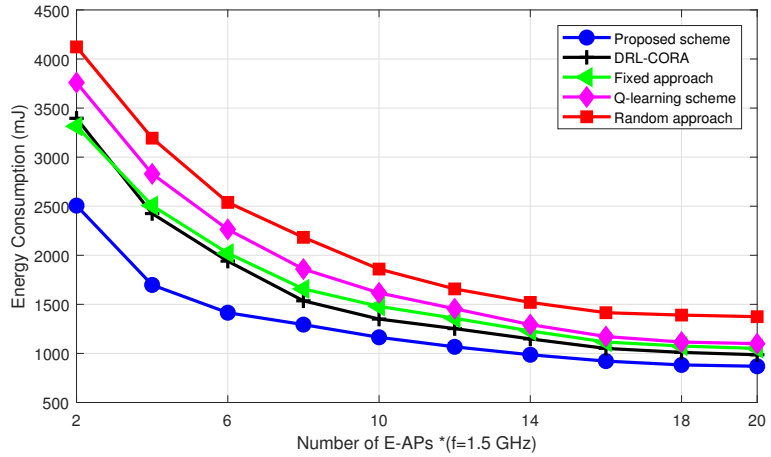

Figure 5. The evaluation of computing energy versus number of E-APs.

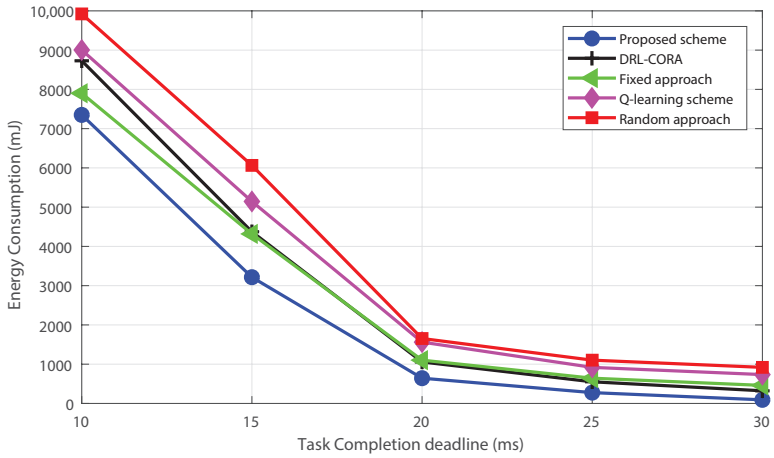

Figure 6. The evaluation of computing energy versus task completion deadline.

As illustrated in Figure 7, interference can also influence the offloading decision and may be influenced by the decisions of other users. Figure 7 highlights the significance of the active user count within the network. As the quantity of users inside the network expanded, there was a corresponding increase in interference, leading to a reduction in the signal-to-interference-plus-noise ratio (SINR) and subsequently a decline in the data rate. This led to higher latency and energy consumption. Therefore, the proportion of local users (not necessarily the exact quantity) also increased. Figure 7 depicts the comparison conducted to examine the influence of interference. For a small number of users, the level of interference was insignificant due to the lower level of interference imposed by others and, therefore, it had no effect on the overall solution. As the quantity of users escalated, the impact of interference became increasingly substantial and could not be ignored. Therefore, considering an interference-free scenario can lead to incorrect decisions.Thus, our proposed solution outperformed the other approaches for all numbers of users.

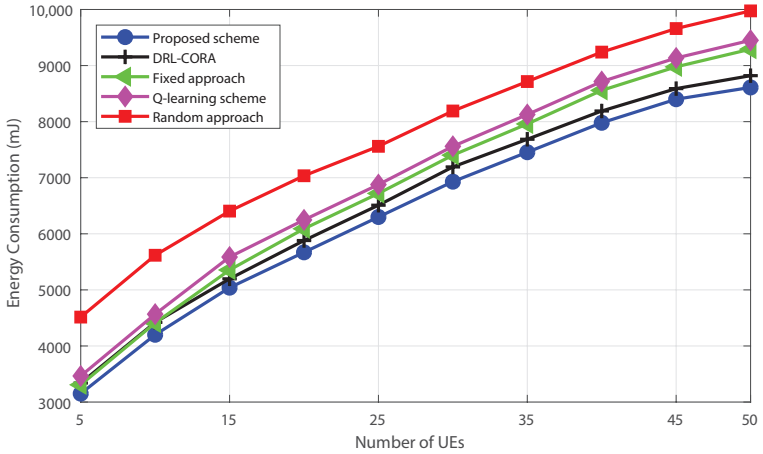

Figure 7. Energy consumption over different numbers of UE.

6. Conclusions

This article presents an analysis of a technique that involves power allocation and cache placement in order to achieve energy efficiency in mobile edge computing networks (MECNs). The minimization of energy consumption is achieved by formulating the energy optimization problem as a MINLFP (mixed integer non-linear fractional programming) problem and subsequently representing it from the standpoint of a Markov decision process (MDP). In order to address the issue at hand, we developed an advanced asynchronous advantage actor-critic (A3C) approach that utilizes multiple agents. This strategy is employed for power allocation and content placement, with the objective of enabling a group of mobile edge computing (MEC) nodes to collaboratively provide the required content at the network edge. By doing so, the need for retrieving content from the faraway cloud is reduced. More specifically, rather than retaining the content from a single e-NB, a cluster of e-NB nodes distribute their content via high-speed connectivity to meet users' demanding content requirements. Through the comprehensive simulation results, we showed our proposed method's significance and performance efficiency. According to the results, the proposed technique achieves an optimum solution and reduces energy usage by roughly 9%, 7%, and 14% compared to the Q-learning, fixed, and random techniques, respectively.

Author Contributions: Conceptualization: I.K., S.R., G.M.S.R., and X.T.; Methodology: I.K., S.R., R.K., and G.M.S.R.; Formal analysis: S.R., R.K., W.u.R., and G.M.S.R.; Investigation: I.K. and G.M.S.R.; Writing—original draft: I.K.; Writing—review and editing: R.K. and W.u.R.; Visualization: I.K.; Supervision: X.T.; Project administration: X.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grant 61932005 and in part by the 111 Project of China under Grant B16006.

Data Availability Statement: The data is an integral part of an ongoing project, and, therefore, it cannot be provided.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhao, J.; Li, Q.; Gong, Y.; Zhang, K. Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks. *IEEE Trans. Veh. Technol.* **2019**, *68*, 7944–7956. [CrossRef]
2. Min, M.; Xiao, L.; Chen, Y.; Cheng, P.; Wu, D.; Zhuang, W. Learning-based computation offloading for IoT devices with energy harvesting. *IEEE Trans. Veh. Technol.* **2019**, *68*, 1930–1941. [CrossRef]
3. Zhu, H.; Cao, Y.; Wei, X.; Wang, W.; Jiang, T.; Jin, S. Caching transient data for Internet of Things: A deep reinforcement learning approach. *IEEE Internet Things J.* **2018**, *6*, 2074–2083. [CrossRef]
4. Peng, M.; Yan, S.; Zhang, K.; Wang, C. Fog-computing-based radio access networks: Issues and challenges. *IEEE Netw.* **2016**, *30*, 46–53. [CrossRef]

5. Peng, M.; Zhang, K. Recent advances in fog radio access networks: Performance analysis and radio resource allocation. *IEEE Access* **2016**, *4*, 5003–5009. [CrossRef]
6. Ceselli, A.; Premoli, M.; Secci, S. Mobile edge cloud network design optimization. *IEEE/ACM Trans. Netw.* **2017**, *25*, 1818–1831. [CrossRef]
7. Zhao, Z.; Bu, S.; Zhao, T.; Yin, Z.; Peng, M.; Ding, Z.; Quek, T.Q. On the design of computation offloading in fog radio access networks. *IEEE Trans. Veh. Technol.* **2019**, *68*, 7136–7149. [CrossRef]
8. Azizi, S.; Othman, M.; Khamfroush, H. DECO: A Deadline-Aware and Energy-Efficient Algorithm for Task Offloading in Mobile Edge Computing. *IEEE Syst. J.* **2022**, *17*, 952–963. [CrossRef]
9. Mao, S.; Wu, J.; Liu, L.; Lan, D.; Taherkordi, A. Energy-efficient cooperative communication and computation for wireless powered mobile-edge computing. *IEEE Syst. J.* **2020**, *16*, 287–298. [CrossRef]
10. Huang, J.; Wan, J.; Lv, B.; Ye, Q.; Chen, Y. Joint Computation Offloading and Resource Allocation for Edge-Cloud Collaboration in Internet of Vehicles via Deep Reinforcement Learning. *IEEE Syst. J.* **2023**, *17*, 2500–2511. [CrossRef]
11. Sartoretti, G.; Paivine, W.; Shi, Y.; Wu, Y.; Choset, H. Distributed learning of decentralized control policies for articulated mobile robots. *IEEE Trans. Robot.* **2019**, *35*, 1109–1122. [CrossRef]
12. Luong, N.C.; Hoang, D.T.; Gong, S.; Niyato, D.; Wang, P.; Liang, Y.C.; Kim, D.I. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3133–3174. [CrossRef]
13. Mach, P.; Becvar, Z. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Commun. Surv. Tutorials* **2017**, *19*, 1628–1656. [CrossRef]
14. Taya, A.; Nishio, T.; Morikura, M.; Yamamoto, K. Deep-reinforcement-learning-based distributed vehicle position controls for coverage expansion in mmWave V2X. *IEICE Trans. Commun.* **2019**, *102*, 2054–2065 [CrossRef]
15. Wang, Z.; Li, M.; Zhao, L.; Zhou, H.; Wang, N. A3C-based Computation Offloading and Service Caching in Cloud-Edge Computing Networks. In Proceedings of the IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), New York, NY, USA, 2–5 May 2022; pp. 1–2.
16. Meng, F.; Chen, P.; Wu, L.; Cheng, J. Power allocation in multi-user cellular networks: Deep reinforcement learning approaches. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 6255–6267. [CrossRef]
17. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
18. Sun, Y.; Peng, M.; Mao, S. Deep reinforcement learning-based mode selection and resource management for green fog radio access networks. *IEEE Internet Things J.* **2018**, *6*, 1960–1971. [CrossRef]
19. Guo, L.; Jia, J.; Chen, J.; Du, A.; Wang, X. Deep reinforcement learning empowered joint mode selection and resource allocation for RIS-aided D2D communications. *Neural Comput. Appl.* **2023**, *35*, 18231–18249. [CrossRef]
20. Wang, Y.; Wang, K.; Huang, H.; Miyazaki, T.; Guo, S. Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications. *IEEE Trans. Ind. Inform.* **2018**, *15*, 976–986. [CrossRef]
21. Chandra, K.R.; Borugadda, S. Multi Agent Deep Reinforcement learning with Deep Q-Network based energy efficiency and resource allocation in NOMA wireless Systems. In Proceedings of the 2023 Second International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), Trichirappalli, India, 5–7 April 2023; pp. 1–8.
22. Babaeizadeh, M.; Frosio, I.; Tyree, S.; Clemons, J.; Kautz, J. Reinforcement learning through asynchronous advantage actor-critic on a gpu. *arXiv* **2016**, arXiv:1611.06256.
23. Zhang, J.; Hu, X.; Ning, Z.; Ngai, E.C.H.; Zhou, L.; Wei, J.; Cheng, J.; Hu, B. Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks. *IEEE Internet Things J.* **2017**, *5*, 2633–2645. [CrossRef]
24. Huynh, L.N.; Pham, Q.V.; Nguyen, T.D.; Hossain, M.D.; Park, J.H.; Huh, E.N. A study on computation offloading in mec systems using whale optimization algorithm. In Proceedings of the 2020 14th International Conference on Ubiquitous Information Management and Communication (IMCOM), Taichung, Taiwan, 3–5 January 2020; pp. 1–4.
25. Waqar, N.; Hassan, S.A.; Mahmood, A.; Dev, K.; Do, D.T.; Gidlund, M. Computation offloading and resource allocation in MEC-enabled integrated aerial-terrestrial vehicular networks: A reinforcement learning approach. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 21478–21491. [CrossRef]
26. Zaman, S.K.u.; Jehangiri, A.I.; Maqsood, T.; Ahmad, Z.; Umar, A.I.; Shuja, J.; Alanazi, E.; Alasmary, W. Mobility-aware computational offloading in mobile edge networks: A survey. *Clust. Comput.* **2021**, *24*, 2735–2756. [CrossRef]
27. He, W.; Wu, S.; Sun, J. An Effective Metaheuristic for Partial Offloading and Resource Allocation in Multi-Device Mobile Edge Computing. In Proceedings of the 2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys), Haikou, China, 20–22 December 2021; pp. 1419–1426.
28. Yuan, X.; Zhu, Y.; Zhao, Z.; Zheng, Y.; Pan, J.; Liu, D. An A3C-based joint optimization offloading and migration algorithm for SD-WBANs. In Proceedings of the 2020 IEEE Globecom Workshops (GC Wkshps), Taipei, Taiwan, 7–11 December 2020; pp. 1–6.
29. Rahman, G.S.; Peng, M.; Yan, S.; Dang, T. Learning based joint cache and power allocation in fog radio access networks. *IEEE Trans. Veh. Technol.* **2020**, *69*, 4401–4411. [CrossRef]
30. Wen, Y.; Zhang, W.; Luo, H. Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones. In Proceedings of the 2012 Proceedings IEEE INFOCOM, Orlando, FL, USA, 25–30 March 2012; pp. 2716–2720.
31. Tuli, S.; Ilager, S.; Ramamohanarao, K.; Buyya, R. Dynamic scheduling for stochastic edge-cloud computing environments using a3c learning and residual recurrent neural networks. *IEEE Trans. Mobile Comput.* **2020**, *21*, 940–954. [CrossRef]

32. Li, Y.; Qi, F.; Wang, Z.; Yu, X.; Shao, S. Distributed edge computing offloading algorithm based on deep reinforcement learning. *IEEE Access* **2020**, *8*, 85204–85215. [CrossRef]
33. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 19–24 June 2016; pp. 1928–1937.
34. Thar, K.; Oo, T.Z.; Tun, Y.K.; Kim, K.T.; Hong, C.S. A deep learning model generation framework for virtualized multi-access edge cache management. *IEEE Access* **2019**, *7*, 62734–62749. [CrossRef]
35. Li, C.; Zhang, Y.; Gao, X.; Luo, Y. Energy-latency tradeoffs for edge caching and dynamic service migration based on DQN in mobile edge computing. *J. Parallel Distrib. Comput.* **2022**, *166*, 15–31. [CrossRef]
36. Rahman, G.S.; Dang, T.; Ahmed, M. Deep reinforcement learning based computation offloading and resource allocation for low-latency fog radio access networks. *Intell. Converg. Netw.* **2020**, *1*, 243–257. [CrossRef]
37. Fan, Z.; Xu, Y.; Kang, Y.; Luo, D. Air Combat Maneuver Decision Method Based on A3C Deep Reinforcement Learning. *Machines* **2022**, *10*, 1033. [CrossRef]
38. Raza, S.; Wang, S.; Ahmed, M.; Anwar, M.R.; Mirza, M.A.; Khan, W.U. Task offloading and resource allocation for IoV using 5G NR-V2X communication. *IEEE Internet Things J.* **2021**, *9*, 10397–10410. [CrossRef]
39. Khan, I.; Tao, X.; Rahman, G.S.; Rehman, W.U.; Salam, T. Advanced energy-efficient computation offloading using deep reinforcement learning in MTC edge computing. *IEEE Access* **2020**, *8*, 82867–82875. [CrossRef]
40. Chen, Z.; Su, X. Computation offloading and resource allocation based on cell-free radio access network. In Proceedings of the 2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 4–6 March 2022; Volume 6, pp. 1498–1502.