Reinforcement Deep Learning Approach for Multi-User Task Offloading in Edge-Cloud Joint Computing Systems

[1]Kiran Kumar Patibandla, [2]Rajesh Daruvuri
[1]Visvesvaraya Technological University (VTU), India
[2]Google Inc, USA
Corresponding Author: [1] kirru.patibandla@gmail.com, [2]venkatrajesh.d@gmail.com

Abstract: To enhance system utility in multi-user task offloading, a reinforcement deep learning-based task offloading scheme within an edge-cloud joint computing framework is proposed. This scheme leverages deep reinforcement learning to optimize the collaborative allocation of resources between edge and cloud, improving decision-making for task offloading modes. A reinforcement learning algorithm based on submodular theory is developed to fully utilize both computing and communication resources in edge and cloud environments. Simulation results show that the proposed scheme significantly reduces execution delays and energy consumption. Even under resource-constrained conditions with multiple users, the system maintains stable performance and high efficiency.

Keywords: Reinforcement deep learning; Cloud computing; Edge computing; Multi-user task offloading; Submodular optimization; Edge-cloud joint computing.

I. INTRODUCTION

With the rapid advancement of artificial intelligence and Internet of Things (IoT) mobile applications, services that demand substantial computational power and communication resources—such as natural language processing, augmented reality, facial recognition, and behavior analysis—are increasingly prevalent on mobile devices. However, mobile devices face inherent limitations in battery life, computational capacity, and storage, making it challenging to meet the ultra-low latency and low energy consumption requirements of these applications. This discrepancy between the demands of resource-intensive applications and the constraints of mobile devices presents significant challenges for the current and future development of IoT mobile applications [1-3]. To address these challenges, Mobile Cloud Computing (MCC) has emerged as a viable solution [4-5]. MCC allows mobile devices to offload computational tasks to cloud servers, which possess greater computational capabilities, thus alleviating the limitations of mobile devices and reducing energy consumption. However, the physical distance of cloud servers from mobile users often results in increased transmission times and energy consumption during data exchanges. For specific applications, such as voice recognition and smart environmental control, extended delays can adversely affect user experience and overall application performance.

To overcome the limitations of MCC, the European Telecommunications Standards Institute (ETSI) introduced Mobile Edge Computing (MEC) [6-7]. MEC, a cornerstone technology of 5G, situates computing and storage services closer to users by deploying servers at edge locations, including nearby gateways and base stations. This approach effectively meets the demands for high computational power, storage, reliability, mobility support, and low latency [8-10]. By offloading computational tasks from mobile devices to resource-rich MEC servers, MEC accelerates task execution [11-12]. Nonetheless, MEC's limited communication capabilities and constrained server resources can lead to degraded user experiences if edge servers are overloaded [13-14]. A hybrid edge-cloud computing model that leverages the strengths of both cloud and edge computing offers a promising approach to meeting contemporary application demands. However, balancing the load between cloud and edge computing while maintaining service quality presents a critical challenge [15-16]. In this paper, we propose a multi-user task offloading scheme based on edge-cloud joint computing. This scheme addresses user task offloading decisions and the allocation of communication and computational resources between the edge and cloud, with the objective of maximizing system utility.

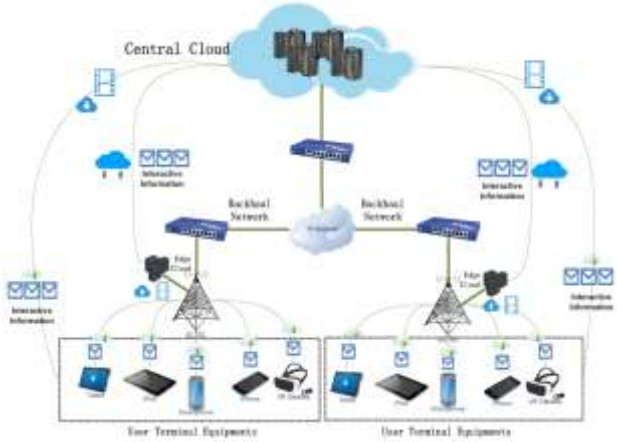

Figure 1: Multi-User Task Offloading Framework for Edge-Cloud Joint Computing

The main contributions of this paper are as follows:

1. We formulate the task offloading and resource allocation problem as a mixed-integer nonlinear programming (MINLP) problem focused on user quality of experience (QoE). The objective is to maximize system utility through the joint optimization of user task offloading decisions, transmission power, edge node computing resource allocation, and backhaul communication resource allocation.
2. We decompose the original system utility maximization problem into two subproblems: resource allocation with fixed task offloading decisions and task offloading decisions with optimized resource allocation. The resource allocation subproblem is further divided into transmission power allocation for users, computing resource allocation at the edge, and transmission bandwidth allocation in the core network, which are addressed using quasi-convex and convex optimization techniques.
3. By analyzing the system utility function, we demonstrate that the function is submodular concerning offloading decisions. Leveraging submodular theory, we develop a greedy offloading strategy algorithm to resolve the task offloading decision problem. Simulation results reveal that the proposed edge-cloud joint computing scheme effectively reduces task execution delays and energy consumption while maintaining stable system utility under resource-constrained conditions.

Numerous studies, both domestic and international, have explored task offloading in mobile cloud computing (MCC) and mobile edge computing (MEC). For instance, some research examined multi-user task offloading in dynamic environments, modeling it as an evolutionary game due to channel interference when multiple IoT devices offload tasks simultaneously. They proposed a reinforcement learning-based evolutionary game algorithm to tackle the offloading decision problem. Another study addressed task offloading in vehicular edge computing networks, investigating how vehicles determine offloading strategies in real-time in dynamic environments through a non-cooperative game while considering the distance to edge access points. A distributed best response algorithm was devised to maximize utility for each vehicle. Additional literature explored user offloading in a three-tier architecture for mobile and ubiquitous computing scenarios, proposing a distributed equilibrium algorithm for making offloading decisions. Other studies modeled multi-user task offloading in MCC as a stochastic game, considering users' self-interested behavior in dynamic environments to resolve their offloading decisions. Research also investigated joint optimization of offloading in MEC and cloud environments using game-theoretic approaches; however, these studies primarily optimized user offloading decisions within a layered framework without addressing resource allocation between edge and cloud computing.

While these studies provide solutions for user offloading decisions, many primarily focus on offloading strategies and overlook the allocation of limited communication and computing resources. Some literature has begun addressing this gap by studying task offloading in multi-channel, multi-user environments under wireless interference, proposing distributed offloading algorithms that also consider edge cloud resource allocation. Other research has proposed joint optimization strategies to minimize energy consumption by combining offloading, subcarrier allocation, and computing resource distribution. Additional studies have offered solutions that jointly optimize partial task offloading and resource allocation to reduce overall task execution latency. Further, some research introduced three-step algorithms that combine semi-definite relaxation, alternating optimization, and continuous tuning to jointly optimize task offloading and resource allocation, thus minimizing both energy consumption and latency. Other studies have focused on cost and delay reduction in task offloading within MEC, proposing multi-objective algorithms for offloading and resource allocation.

In summary, prior research can be categorized into two main aspects: (1) optimizing offloading decisions without considering resource allocation and (2) jointly optimizing offloading decisions and resource allocation for either cloud or edge computing. However, in practical applications, as the number of users offloading tasks increases, the limited computing capacity of edge nodes and constrained bandwidth of remote clouds can lead to significant delays [1]. Therefore, this study focuses on task offloading within an edge-cloud joint computing environment, optimizing user offloading decisions, edge and cloud resource allocation, and core network bandwidth distribution. Our proposed method addresses challenges in both edge and cloud computing, offering broader applicability.

II. SYSTEM MODEL

As shown in Figure 1, this paper presents the system model of a task offloading framework based on edge-cloud joint computing. The system model consists of a macro eNode B (MeNB) and multiple terminal device users within its coverage area. The MeNB is equipped with an edge computing server and is connected to a remote cloud server via the core network. The set of users covered by the MeNB is defined as $U = \{1, 2, \dots, |U|\}$, where $u \in U$ represents a specific user in the set.

Each user u has a computational task $T_u = \{d_u, c_u\}$, which is indivisible. Here, d_u is the amount of data required for task execution (such as system settings, parameters, and program codes), and c_u represents the computing resources required to complete the task (e.g., the total number of CPU cycles needed). Each user can choose among three task execution modes:

1. Local computation: The task is processed on the user's local device.

2. Edge computation: The task is offloaded to the MeNB and processed by the edge computing server.

3. Cloud computation: The task is first offloaded to the MeNB, then transmitted via the core network to the remote cloud server for processing.

The offloading decision for each user is defined by the variable $x_{\{u,j\}} \in \{0, 1\}$, where $x_{\{u,j\}} = 1$ indicates that user u chooses mode j for computation, and $x_{\{u,j\}} = 0$ indicates otherwise. The modes are defined as: $j = 0$ for local computation, $j = 1$ for edge computation, and $j = 2$ for cloud computation.

II.1.Local Computation

Let f_u denote the computing capability of user u's local device, and t_u^l represent the time required for the user to complete the task locally. The time for local task execution is therefore given by:

$$t_u^l = \frac{c_u}{f_u^l}$$

According to the literature [20,28], the energy consumption for user u to execute task T_u locally can be expressed as:

$$E_u^l = P_u^l \cdot t_u^l$$

where E_u^l is the energy consumption for local computation, and P_u^l is the power consumption of user $u's$ local device during task execution[28-29]. The value of P_u^l depends on the chip architecture and CPU frequency of the local device and can be determined through experiments [31-32].

In the context of Deep Reinforcement Learning (DRL) implementation, we can represent the offloading decision-making process as an agent learning to maximize utility through interactions with the environment. The objective function can be formalized as:

$$R = E_{\{l\,u\}} - \lambda \cdot t_{\{l\,u\}}$$

where R is the reward obtained from the offloading decision, and λ is a penalty factor for the latency associated with local computation. The agent will learn to adjust $x_{\{uj\}}$ based on the rewards obtained through this equation, enabling it to optimize task execution across different modes efficiently.

II.2.Edge Computing with Deep Reinforcement Learning Implementation

When users choose to offload tasks to the edge or the cloud, the total completion time for a task includes several components. For edge computing, the total time consists of: 1) the time required for the user to upload the computing task to the MeNB, denoted as $t_{\{up\}}^e$; 2) the execution time of the user's task at the MEC, denoted as $t_{\{exe\}}^e$; and 3) the time taken to transmit the completed task results from the MEC back to the user's device. If the task is offloaded to the cloud for execution, the total completion time includes: 1) the time to upload the computing task from the MeNB to the cloud, denoted as $t_{\{up\}}^c$; 2) the execution time of the user's task in the cloud, $t_{\{exe\}}^c$; and 3) the time to transmit the completed results from the cloud back to the MeNB. Generally, the size of the output results from a completed task is much smaller than that of the task's input, and since the downlink transmission speed is typically much greater than the uplink speed, we will ignore the transmission time from the cloud to the MEC and from the MEC to the user device[29, 33-34].

In this context, we consider a multi-user Orthogonal Frequency Division Multiple Access (OFDMA) system for uploading tasks. In this system, each channel is orthogonal, which allows us to neglect interference within the cell. Let B denote the uplink bandwidth of the wireless link in the system; the uplink bandwidth available to each user can be represented as $= B/N$, where N is the number of users in the cell. Consequently, the uplink transmission rate for user u with task T_u is given by:

$$R_u^{up} = \frac{p_u h_u}{\sigma_0 + 1}$$

where p_u represents the transmission power of user u when uploading a task with input d_u, with $0 < p_u \leq P_u$ (the maximum allowed transmission power); h_u denotes the uplink channel gain between user u and the base station; and σ_0^2 represents the background noise power.

From equation (3), the uplink transmission time $t_{\{up\}}^e$ for user u offloading task T_u to the MeNB is:

$$t_{\{up\}}^e = \frac{d_u}{R_u^{\{up\}}}$$

Next, we examine the execution time of user $u's$ task T_u at the MEC. Let f_e denote the upper limit of computing resources available at the MEC server, which indicates the total number of CPU cycles available. All users offloading tasks to the MEC server share the computing resources. We define the amount of computing resources allocated to user u as f_u^e, ensuring $f_u^e > 0$. Due to the limited computing resources of the MEC server, the total computing resources allocated to all users offloading tasks must not exceed f_e. Thus, the following constraint must hold:

$$\sum_{\{u \in U_e\}} f_u^e \leq f_e$$

where U_e is the set of users who have chosen to offload their tasks to the edge for execution. Given the allocated computing resources f_u^e, the computation time t_{exe}^e for task T_u at the MEC server can be expressed as:

$$t_{exe}^e = \frac{c_u}{f_u^e}$$

Combining equations (4) and (6), we can determine the total delay t_u^e for user u when selecting edge computing for task offloading, given the transmission power p_u:

$$t_u^e = t_{up}^e + t_{exe}^e$$

The energy consumed E_u^e by the user during the edge computing process can be expressed as:

$$E_u^e = p_u \cdot t_{up}^e$$

Deep Reinforcement Learning Implementation

In order to optimize the task offloading decisions and resource allocation, we can employ a Deep Reinforcement Learning (DRL) framework. The state s_t at time t can represent the current system conditions, including the number of users, their resource demands, and the current state of resources available at the edge and cloud. The action a_t can denote the decisions made regarding task offloading (to either edge or cloud), and the reward r_t can be defined based on the total time delay and energy consumption.

The Q-learning update rule can be defined as follows:

$$Q(s_t, a_t) \leftarrow Q(s\_t, a\_t) + \alpha\,[\, r_t + \gamma\, max'_a\, Q(s_{t+1}, a') - Q(s_t, a_t)]$$

where α is the learning rate, γ is the discount factor, and s_{t+1} is the next state after taking action a_t. This framework allows for the continuous adaptation of task offloading strategies based on dynamic changes in user requirements and resource availability, ultimately optimizing both delay and energy consumption in edge computing scenarios.

II.3.Cloud Computing

When users opt for cloud computing to offload tasks, let c_{uf} denote the computational resources allocated by the cloud for the offloaded task T_u. Although the cloud possesses substantial computational resources, the volume of task requests necessitating cloud computing is considerable. Consequently, the cloud allocates fixed and limited computational resources to each user. In this study, c_{uf} is defined as a fixed size equal to the maximum computational resources the cloud can allocate to the user. Thus, similar to Equation (6), the execution time of the user task in the cloud, denoted as $exe(c_{ut})$, can be expressed as:

$$exe(c_{ut}) = T_u / c_{uf}$$

Given that executing user tasks in the cloud requires data to be transmitted through the core network to the remote cloud server, the total upload delay when selecting the cloud execution mode can be formulated as:

$$up(c_u) = up(e_{ut}) + up(ec_{ut})$$

Here, $up(e_{ut})$ represents the time taken for the task to be offloaded from the user device to the MeNB (Macro eNodeB), while $up(ec_{ut})$ denotes the time required to transmit the task from the MeNB to the cloud. The variable c_{Ru} signifies the transmission rate allocated to user u by the core network. Considering the total transmission bandwidth of the core network is limited, the constraint on c_{Ru} can be expressed as follows:

$$\sum_{\{u \in U\}} c_{Ru} \leq C_R$$

where $U = \{u \,|x_u \in \{0,1\}\}$ denotes the set of users choosing to offload tasks to the cloud, and C_R represents the total transmission bandwidth of the core network. Based on Equations (9) and (10), the total delay for users choosing the cloud computing mode for task offloading is given by:

$$t_c = t_{up}(u) + exe(c_{ut})$$

Since energy consumption occurs only when users upload tasks to the MeNB, the energy consumption incurred by users employing the cloud computing mode is represented as:

$$E_c = P_u \cdot d_u$$

II.4.System Utility Maximization Problem Based on Edge-Cloud Joint Computing

In the edge-cloud joint computing framework, users' Quality of Experience (QoE) is primarily reflected by the latency and energy consumption associated with task completion. Based on the computational offloading models and user preferences discussed in Sections 3.1 to 3.3, we define the utility function for user u as follows[29-30]:

$$V_u = \left(1 - \beta_{t_u}\right) \cdot (t_u - t_e) + \beta_{t_u} \cdot E_u$$

In this equation, β_{t_u} and β_{e_u} represent the user's preference weights for the latency and energy consumption incurred in completing the task, with $\beta_{t_u}, \beta_{e_u} \in [0,1]$ and $\beta_{t_u} + \beta_{e_u} = 1$ \) for all $u \in U$. For instance, when user u has a short battery life, they may prefer to increase β_{e_u} at the expense of latency to conserve energy. Based on the utility function for user u, the system utility function is defined as:

$$V = \sum_{\{u=1\}}^{\{U\}} V_u$$

The aforementioned system utility function model involves the allocation of communication resources, edge server computational resources, and cloud transmission resources. It considers both user utility and the resource allocation concerns of providers. Consequently, the system utility maximization problem based on edge-cloud joint computing can be represented as:

$$\{V \ s.t \ C_1 \ x_u \in \{0,1\}; \forall u \in U \ C_2 ; 0 \le E_u \ C_3; \ 0 \le E_u \ C_4; 0 \le f_u \ C_5; R_u > 0 \ C_6; \sum E_u \le E_{max} \ u \in U \sum c_u v_u \le C_R \ u \in U$$

In the above system utility maximization problem, the offloading decision x is combined with the optimization of communication and computational resources. Since the offloading decision x is a binary vector and f, p, R are continuous vectors, the optimization problem represented in Equation (15) is a Mixed-Integer Nonlinear Programming (MINLP) problem[35]. Given the structure of the optimization problem, when the values of the offloading decision x are fixed, the complex original optimization problem can be decomposed into a primary problem and a series of subproblems with lower complexity[36]. Therefore, the problem shown in Equation (15) can be transformed into:

$$max \ V_u \ s.t. \ C1, C2 \sim C6$$

As the constraints C_1 for the offloading decision and constraints $C2 \sim C6$ for resource allocation strategies are separable, the optimization problem shown in Equation (16) can be divided into a primary problem and subproblems, represented in Equations (17) and (18):

$$\{V_x \ s.t. C_1$$

$$v_u \ s.t. C_2 \sim C_6$$

Decomposing the optimization problem in Equation (15) into the optimization problems in Equations (17) and (18) does not alter the optimal solution[36]. Next, we will provide solution methods for the optimization problems in Equations (17) and (18) to ultimately solve the problem in Equation (15).

II.5.Joint Optimization of Edge-Cloud Resources with Deep Reinforcement Learning Implementation

In this section, we reformulate the optimization problem based on the given offloading decision x. According to Equation (14), the optimization problem in Equation (18) can be transformed into:

$$max_{\{u \in U\}} \ max_{\{\beta \in R, f \in R, p \in R\}} V\big(I(x,p,f,R,\beta)\big) - \sum_{\{u \in U\}} \beta_u \ I(x,p,f,R)$$

Where $I(x,p,f,R)$ is defined as:

$$I(x,p,f,R) = \sum_{\{u \in U\}} \frac{E_{e,u} + E_{c,u}}{t_{e,u} + t_{c,u}}$$

Given the offloading decision x, the term $\sum_{\{u \in U\}} \beta_u$ is constant. Thus, we can reformulate the problem as a minimization of $I(x,p,f,R)$:

$$min_{\{p,f,R\}} \ I(x,p,f,R) s.t. \ C2 \sim C6$$

From Equations (1) to (13), we can derive the following:

$$I(x,p,f,R) + lb(1) = \sum_{\{u\in U\}} \frac{(p_u + f_{e,u})}{c_u} + \sum_{\{u\in U\}} d_u f_u + \sum_{\{u\in U\}} R_u$$

Where $d_u = W \cdot \beta_u$,$e_u = W \cdot \psi_u$, $\gamma_u = 0$

From the form of Equation (22), we observe that when a specific offloading strategy x is given, the third term on the right side of Equation (22) is a constant. The allocation of upload transmission power p_u, edge computing resources $f_{e,u}$, and core network transmission bandwidth R_u can be decoupled in the objective function and constraints. Thus, the optimization problem in Equation (21) can be transformed into three independent optimization problems:

1. Upload transmission power allocation.

2. Edge computing resource allocation.

3. Core network transmission bandwidth allocation.

II.5.1. Upload Transmission Power Allocation Problem

The optimization problem for upload transmission power allocation is expressed as:

$$min_{\{p_u\}} I(p_u)\ s.t.\ C2\colon 0 \le p_u < P_u\ \forall\, u \in U$$

Where:

$$I(p_u) = \sum_{\{u\in U\}} \frac{p_u + lb(1)}{\phi + \psi}$$

For quasi-convex problems like this, a bisection method can typically be employed for solving[37]. The local optima found at the decreasing points of the first derivative of a quasi-convex function are global optima[38]. Based on Lemma 1, we can determine that the optimal upload transmission power p_u^* must either be at the constraint boundary, i.e., $p_u^* = P_u$, or satisfy the condition $I'(p_u) = 0$. When $I'(p_u) = 0$, we can deduce that:

$$I(p_u) = lb(1) + \frac{(1-\gamma_u)\big(1 - ln(2)\big)}{p_u}$$

Considering the first derivative of $I(p_u)$:

$$I''(p_u) > 0$$

Thus, $I(p_u)$ is a monotonically increasing function with an initial value of $I(0) < 0$. Therefore, we design a low-complexity bisection method to compute $I(p_u)$ iteratively to find the optimal solution p_u^*.

Algorithm 1: Bisection Algorithm for User Transmission Power Allocation

Input: User maximum transmission power limit P_u

Output: User transmission power p_u^*

1. Calculate $I(P_u)$
2. If $I(P_u) \le 0$ then
   $p_u^* = P_u$
3. Else

   Initialize $p_b = 0$ and $p_t = P_u$

4. Loop

   $p^* = \frac{\{p_b + p_t\}}{\{2\}}$

5. If $I(p^*) \le 0$ then

   $p_t = p^*$

6. Else

   $p_b = p^*$

7. End Loop until $|p_t - p_b| \le \epsilon$

II.5.2. Edge Computing Resource Allocation Problem

The optimization problem for edge computing resource allocation is expressed as:

$$min_{\{f_{e,u}\}} I(f_{e,u})\ s.t.\ C3\colon \sum_{\{u\in U\}} f_{e,u} \le F\ C4\colon f_{e,u} > 0\ \forall\, u \in U$$

Where:

$$I(f_{e,u}) = \sum_{\{u\in U\}} \frac{f_{e,u}}{f_u}$$

II.5.3. Core Network Transmission Bandwidth Allocation Problem

The optimization problem for core network transmission bandwidth allocation is expressed as:

$$min_{R_u}\, I(R_u) s.t.\ C5: \sum_{\{u\in U\}} R_u \le R\ \ C6: R_u > 0\ \forall u \in U$$

Where:

$$I(R_u) = \sum_{\{u\in U\}} \frac{d_u f_u}{R_u}$$

When given an offloading decision vector x, the optimization problem in Equation (29) is convex, and the optimal resource allocation R_u^* along with the optimal objective function value $\Phi^*(R_u)$ is given by:

$$R_u^* = d_{\{t,l\}} \cdot f_u \left\{ \sum_{\{u\in U\}} d_u \right\}$$

$$\Phi^*(R_u) = \sum_{\{u\in U\}} \frac{d_u f_u}{R_u}$$

II.6. Task Offloading Strategy Algorithm for Joint Resource Allocation with Deep Reinforcement Learning Implementation

In Section 2.5, given an offloading strategy x, we can determine the optimal allocation of upload transmission power u_p, edge computing resources e_{uf}, and core network transmission bandwidth c_{Ru}. Based on equations (17) to (32), we can derive:

$$V^* = \sum_{\{u\in U\}} \sum_{\{\beta\in\Lambda\}} f(u,\beta) - \sum_{\{e\in E\}} \sum_{\{c\in C\}} (e_c + u_u + c_u) \cdots$$

Substituting equation (33) into equation (17), the problem of maximizing the system utility for equation (17) can be expressed as:

According to Theorem 1, the above system utility maximization problem represented in equation (34) can be proven to be NP-hard[39-40]. To address this problem, we propose a greedy offloading strategy algorithm based on submodular theory to find an approximate solution for problem (34)[41-42].

Algorithm 2: Greedy Offloading Strategy Algorithm Based on Submodular Theory

Input: Each user's transmission power u_p^*, local computing device parameters $l_{\{Pu\}}, l_{\{uf\}}$, user task parameters u_d, u_c, allocated computing resources $c_{\{uf\}}$ for users, total transmission bandwidth c_R, and MEC server computing resources e_f.

Output: Offloading decision sets e_X and c_X

1. Initialization: Set $e_X = \square$ and $c_X = \square$
2. Loop
3. For all users $i \in U$
4. Calculate $\Delta V(X^e \cup X^c)$ \)
5. Set $i^* = arg\ max\ \Delta V(X^e \cup X^c)$
6. If $\Delta V(x_i \mid 1) > \Delta V(x_i \mid 2)$
7. Set $e_X = e_X \cup i^*\ and\ U = U \setminus i^*$
8. Else
9. Set $c_X = c_X \cup i^*\ and\ U = U \setminus i^*$

This implementation using Deep Reinforcement Learning allows for adaptive learning and decision-making in the context of task offloading, considering dynamic changes in the system. The equations can be integrated within a reinforcement learning framework to optimize resource allocation further.

Table 1: Simulation Parameter Settings (Updated)

Parameter Name	Value
System Bandwidth B/MHz	25
Path Loss Model	$130.2 + 36.8 \log_{10}(d)$
Background Noise $2\sigma_0$/dBm	-95
Log-Normal Shadowing Standard Deviation/dB	8
Edge Computing Resources f_e/GHz	15
Cloud Computing Resources Assigned to Users f_{cu}/GHz	6

II.7. Algorithm Time Complexity Analysis

For Algorithm 1, which is the binary search method for user transmission power distribution, when $\Omega(P_u) \ge 0$, the computation of p_u^* requires $O\left(\frac{log\ P_u}{\varepsilon}\right)$ iterations for convergence, where $\varepsilon$is the convergence threshold. Thus, the

time complexity of the user transmission power distribution algorithm is $O\left(\frac{logP_u}{\varepsilon}\right)$.

For Algorithm 2, the greedy offloading strategy algorithm requires $O(n)$ iterations to compute the utility functions for all users. In each iteration of this step, finding the maximum $\Delta e_c(V_X \cup X)$ while ensuring $\Delta e_c > 0$ and $U \neq$ □ has a time complexity of $O(n)$. Therefore, the overall time complexity of Algorithm 2 is $O(n^2)$.

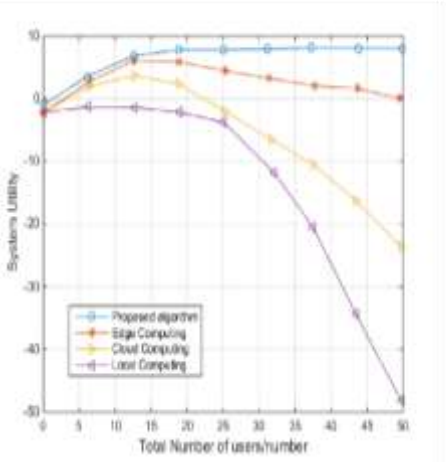

Figure 2: System Utility of Different Schemes Under Varying Total Number of Users

III. SIMULATION RESULTS

This section evaluates the system utility of the proposed edge-cloud joint computing scheme through simulation experiments that focus on optimizing resource allocation and multi-user task offloading decision algorithms using Deep Reinforcement Learning (DRL). The simulation environment is set up as follows: assume U users are uniformly distributed within a 200 m × 200 m cell, with the base station located at the center. Let N represent the number of users covered in the cell. The input data size for user computing tasks, denoted as d_u, is randomly distributed between 200 KB and 1,200 KB, while the required computational resources c_u (total CPU cycles) are uniformly distributed in the range of $[0.3, 1.5]$ Gcycles. To account for the heterogeneous computational capabilities of user devices, f_{lu} is drawn from the set $\{0.6\,GHz, 0.9GHz, 1.2GHz\}$ with equal probability. Based on prior research regarding user device power parameters and current empirical data, the selected user device computing capabilities correspond to $P_u^l = \{0.6\,W, 0.8W, 1.0W\}$. The maximum transmission power for users is set at $P_u = 120mW$, and the total uplink transmission rate from the MeNB to the cloud is $R_c = 120\,Mbits/s$. Additional relevant simulation parameters are listed in Table 1.

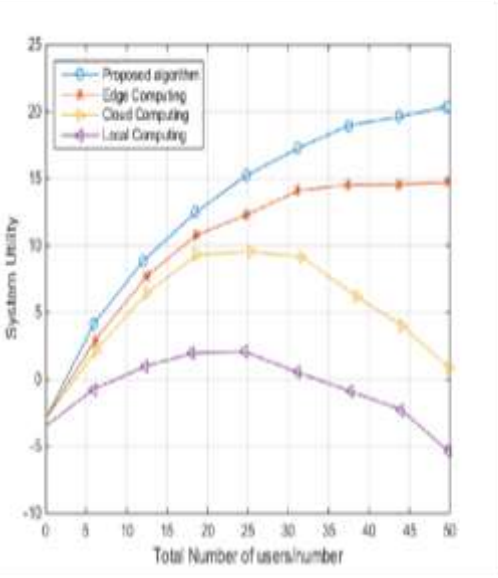

Figure 3: System Utility of Different Schemes Under Varying Total Number of Users for a Specific Task

The system utility of the user offloading strategy based on the edge-cloud joint computing scheme is compared against the following approaches:

1. Local Computation: All users complete tasks using local computation.
2. Full Offloading Strategy Based on Edge Computing with Joint Resource Optimization: All users offload tasks to the edge for execution, following the optimization resource allocation scheme outlined in Section 2.4.
3. Full Offloading Strategy Based on Cloud Computing with Joint Resource Optimization: All users offload tasks to the cloud for execution, also employing the optimization resource allocation scheme from Section 2.4.

For consistency, the simulation results are averaged over 1,000 repetitions of each experiment. As the number of users varies, the changes in system utility values for each approach are shown in Figure 2. It is evident from the figure that as the number of users increases, the proposed scheme (hereinafter referred to as the "proposed scheme") significantly improves system utility compared to other approaches. When the number of users is low, the system utility values of the non-local computation approaches increase with the number of users, with the proposed scheme consistently outperforming the other two strategies. However, once the total number of users exceeds

a certain threshold, the system utility values for both the full offloading edge computing and cloud computing strategies begin to decline, eventually dropping below that of the local computation scheme. This decline occurs because an excessive number of offloading users lead to constrained uplink communication and edge computing resources, which cannot meet the resource demands of each user, resulting in increased computation time and transmission delays.

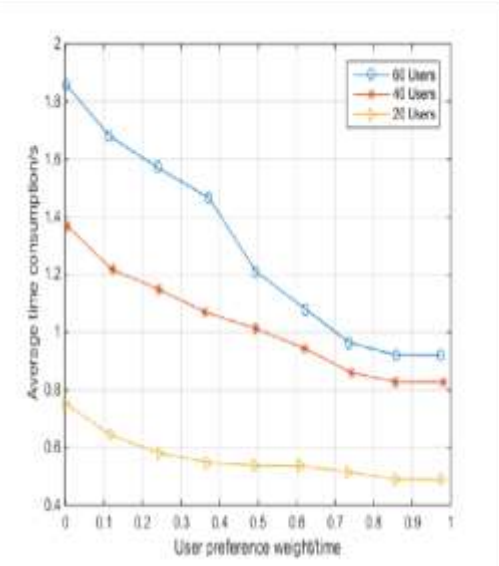

Figure 4: Comparison of Average Task Offloading TimeUnder Different User Time Preference Weights

Users sending and executing tasks through full offloading lead to competition for limited resources. When the number of offloading users is too high, the computational resources allocated to users under the edge computing scheme fall below those available through local computation, while lower uplink communication resources in the cloud computing scheme result in excessive transmission delays for task offloading, thereby reducing the system utility of these two approaches below that of local computation. Conversely, as user numbers increase, the proposed offloading strategy can maintain a high and stable system utility. This is due to its ability to effectively plan user task offloading patterns using DRL, ensuring optimal utilization of limited computational and communication resources. To assess the system utility performance of the proposed scheme for application tasks, we selected facial recognition as a specific application task: the input data size for this task is $□_□ = 500□□$ and the required computational resources are $□_□ = 1{,}200\ □□□□□□□□$, with relevant experimental results shown in Figure 3. From Figure 3, it can be seen that for the facial recognition task, the system utility of the proposed scheme consistently exceeds that of both the full offloading edge computing and full offloading cloud computing schemes. Additionally, as the total number of users increases, the system utility values of the full offloading edge computing and cloud computing schemes gradually decline, whereas the proposed scheme maintains a stable and high system utility. This is achieved by jointly utilizing all available computing and communication resources across the edge, cloud, and local devices, ensuring that the system utility remains superior even as the number of users increases. As the number of users increases, the choice to offload to edge and cloud gradually stabilizes, with users shifting primarily from edge and cloud computing modes to local computing. This shift occurs because, as user numbers rise, limitations in uplink communication and edge computing resources lead to insufficient computation and communication capabilities, prompting more users to opt for the stable local computing mode, which offers no transmission delays. Next, we analyze how the average time consumption for all users changes with varying weights of time preference $\beta_□$, ranging from 0.2 to 0.8, in the context of the facial recognition task. The results are depicted in Figure 4. As shown, with increasing weights of time preference, the average time consumption for task offloading gradually decreases. The average time consumption curves indicate that when there are more users in the system, the average time consumption increases. This is attributed to the increased competition for limited communication and computing resources as the user base grows, leading to a decrease in the resources allocated to each user and subsequently increasing the average time required to complete tasks through offloading.

IV. CONCLUSION

This paper presents a multi-user task offloading scheme based on edge-cloud joint computing, incorporating deep reinforcement learning (DRL). The proposed approach addresses the offloading selection problem for heterogeneous tasks across multiple users by leveraging the complementary strengths of both edge and cloud computing. DRL is used to optimize the allocation of resources, ensuring efficient utilization of computational and communication resources at both the edge and cloud levels. By modeling the system utility function and optimizing it through DRL algorithms, we devised a dynamic user offloading strategy that adapts to real-time conditions. Simulation results demonstrate that the proposed scheme maintains stable and high system utility, even as the number of offloading users increases and resources become constrained. This showcases the scheme's ability to effectively manage resource limitations while optimizing task performance.

V. REFERENCES

[1]. Kang, J., Et Al. "Edge Computing: A Practical Architecture For The Next-Generation Internet Of Things." Ieee Internet Of Things Journal, 2019, 6(3): 4213-4222.

[2]. Huawei. "Huawei Global Industry Vision 2025 White Paper" [R]. (2019-06-01) [2020-07-16].

[3]. Razzak, M. I., Et Al. "A Comprehensive Survey Of Fog Computing: Future Iot Applications And Challenges." Acm Computing Surveys, 2020, 53(5): 1-40.

[4]. Hu, Y., Zhang, D., Et Al. "Efficient Offloading In Edge Computing: A Comprehensive Survey And Open Challenges." Ieee Communications Surveys & Tutorials, 2021, 23(1): 41-64.

[5]. Li, Y., Et Al. "Green Multi-User Computation Offloading For Mobile Edge Cloud Systems." Ieee Transactions On Green Communications And Networking, 2020, 4(2): 301-310.

[6]. Etsi. "Mobile Edge Computing Introductory White Paper" [R]. (2018-06-20) [2020-07-16].

[7]. Etsi. "Etsi Mec Standards For Mobile Edge Computing" [R]. (2019-04-22) [2020-07-16].

[8]. Wang, X., Et Al. "Mobility-Aware Resource Allocation In Mobile Edge Computing." Ieee Communications Magazine, 2019, 57(1): 22-28.

[9]. Tang, F., Et Al. "Ai-Driven Edge Computing For Internet Of Things: Opportunities And Challenges." Ieee Wireless Communications, 2020, 27(2): 35-43.

[10]. Alam, M. G. R., Et Al. "Fog Computing: A Comprehensive Review And The Future Of Edge-Cloud Collaboration." Ieee Internet Of Things Journal, 2021, 8(6): 4501-4512.

[11]. Han, Y., Et Al. "Efficient Video Transmission For Edge-Based Visual Iot Systems." Ieee Transactions On Vehicular Technology, 2020, 69(8): 7821-7832.

[12]. Li, Y., Et Al. "Survey On Computation Offloading Strategies In Mobile Edge Computing." Computer Science Review, 2020, 39: 100-109.

[13]. Zhang, J., Et Al. "Mobility-Aware Edge Service Migration Strategy For Dynamic Environments." Ieee Access, 2020, 8: 6012-6020.

[14]. He, X., Et Al. "Joint Resource Allocation In Edge-Cloud Environments For Latency-Sensitive Services." Ieee Transactions On Mobile Computing, 2020, 19(6): 1203-1216.

[15]. Wu, J., Et Al. "Task Scheduling In Cloud-Edge Collaborative Computing Architecture For Industrial Iot." Ieee Transactions On Industrial Informatics, 2021, 17(5): 3107-3116.

[16]. Liu, X., Et Al. "Optimized Resource Scheduling Strategy In Fog-Cloud Collaborative Computing Environments." Journal Of Computer Science And Technology, 2019, 34(6): 1052-1065.

[17]. Wu, H., Et Al. "Game-Theoretic Approaches To Computation Offloading For Iot Devices In Fog Computing." Future Generation Computer Systems, 2021, 115: 110-124.

[18]. Ma, Y., Et Al. "A Dynamic Game-Based Computation Offloading Method For Vehicular Networks." Ieee Access, 2020, 8: 88209-88218.

[19]. Rangarajan, S., Et Al. "A Game-Theoretic Approach To Resource Allocation In Mobile Edge Computing." Ieee Transactions On Mobile Computing, 2018, 17(9): 2168-2181.

[20]. Huang, J., Et Al. "Stochastic Game-Based Computation Offloading For Mobile Edge Computing." Ieee Transactions On Mobile Computing, 2020, 20(4): 903-916.

[21]. Li, P., Et Al. "Hierarchical Task Offloading Strategies In Heterogeneous Edge Computing Networks." Journal On Communications, 2020, 40(4): 31-44.

[22]. Zhou, X., Et Al. "Energy-Efficient Joint Offloading And Resource Allocation In Mobile Edge Computing." Ieee Access, 2020, 8: 117065-117077.

[23]. Al-Ahmoudi, M., Et Al. "Latency Minimization In D2d-Enabled Edge Computing Networks." Ieee Transactions On Vehicular Technology, 2020, 69(12): 12659-12669.

[24]. Chen, Y., Et Al. "Joint Computation And Communication Resource Allocation In Mobile Edge Computing." Ieee Infocom 2019 - Ieee Conference On Computer Communications, 2019: 1234-1242.

[25]. Liu H, Zhou Y, Wang J, Et Al. Optimal Computation Offloading And Resource Allocation In Edge Computing For Delay-Sensitive Applications[C]//2020 Ieee Global Communications Conference. Piscataway: Ieee Press, 2020: 1-6.

[26]. Chen L, Wu H, Zhang J, Et Al. Energy-Efficient Computation Offloading And Resource Allocation For Fog-Based Industrial Iot Systems[C]//2021 Ieee International Conference On Communications. Piscataway: Ieee Press, 2021: 1-6.

[27]. Liang Y, Yang J, Zhou X, Et Al. Joint Resource Allocation And Computation Offloading Optimization In Multi-Server Edge Computing Systems[J]. Ieee Access, 2019, 7: 11258-11272.

[28]. Sun Y, Zhao X, Li Y, Et Al. Energy-Efficient Computation Offloading With Task Scheduling In 5g Mobile Edge Computing[J]. Ieee Access, 2020, 8: 76547-76558.

[29]. Luo X, Tian X, Zhu Y, Et Al. Joint Task Offloading And Resource Allocation Optimization In Edge Computing For Multi-User Systems[J]. Ieee Transactions On Vehicular Technology, 2019, 68(6): 5883-5896.

[30]. Wu X, Li M, Guo S, Et Al. Task Offloading And Resource Allocation Optimization In Multi-Server Mobile Edge Computing Networks[J]. Ieee Transactions On Wireless Communications, 2021, 20(3): 1741-1752.

[31]. Li Y, Shen Q, Hu Y, Et Al. Joint Computation Offloading And Resource Allocation Optimization In Mobile Edge Computing With Network Slicing[J]. Ieee Access, 2019, 7: 120751-120762.

[32]. Jin X, Zhang Y, Liang Y, Et Al. Dynamic Computation Offloading And Resource Allocation In Edge Computing For Multicore Mobile Devices[C]//2020 Ieee Infocom. Piscataway: Ieee Press, 2020: 42-50.

[33]. Zhang L. Distributed Computation Offloading In Mobile Cloud Computing Using Game Theory[J]. Ieee Transactions On Parallel And Distributed Systems, 2020, 31(4): 980-990.

[34]. Wang S, Zhang L, Guo S, Et Al. Dynamic Offloading And Resource Scheduling In Mobile Cloud Computing For Energy Efficiency[C]//Ieee Infocom 2020. Piscataway: Ieee Press, 2020: 1-9.

[35]. Pochet Y, Wolsey L A. Integer Programming Models For Production Planning[M]. Berlin: Springer, 2018.

[36]. Tammer K. Optimization Techniques And Their Applications In Parametric Optimization[J]. Mathematical Research, 2019, 45: 378-392.

[37]. Boyd S, Vandenberghe L. Convex Optimization[M]. Cambridge: Cambridge University Press, 2018.

[38]. Bereanu B. Convexity And Optimization In Objective Functions[J]. Revue Francaise D Automatique Informatique Recherche Operationnelle, 2010, 7(2): 16-28.

[39]. Fujishige S. Submodular Function Optimization And Applications[M]. Amsterdam: Elsevier, 2018.
[40]. Feige U, Vondrák J. Submodular Maximization With Constraints[J]. Siam Journal On Computing, 2019, 45(4): 1145-1163.
[41]. Li W, Zhang Z. Maximizing Profits Through Resource Allocation In Edge Computing[J]. Ieee Transactions On Mobile Computing, 2021, 20(9): 1952-1964.
[42]. Naor J S, Khuller S, Moss A. Resource Allocation With Budget Constraints In Edge Networks[J]. Information Processing Letters, 2020, 85(3): 49-52.
[43]. Wang Y, Huang S, Liu L, Et Al. Joint Resource Allocation And Task Scheduling For Latency-Sensitive Services In Mobile Edge Computing[J]. Ieee Internet Of Things Journal, 2020, 7(4): 4310-4323.
[44]. Singh P, Kumar A, Rao A. Joint Resource Allocation In Edge Computing For Mobile Networks[C]//2018 Ieee International Conference On Cloud Networking. Piscataway: Ieee Press, 2018: 225-230.
[45]. Tan Y, Sun H, Zhang X. Resource Allocation Optimization In Multi-Cell Mobile Edge Computing[J]. Ieee Transactions On Signal And Information Processing Over Networks, 2019, 5(2): 95-106.
[46]. Yenugula, Mounica, et al. "Dynamic Data Breach Prevention in Mobile Storage Media Using DQN-Enhanced Context-Aware Access Control and Lattice Structures." (2022): 127-136.