D2D-Assisted Multi-User Cooperative Partial Offloading in MEC Based on Deep Reinforcement Learning

Xin Guan 1, Tiejun Lv 1, Zhipeng Lin 2,*, Pingmu Huang 3 and Jie Zeng 4

1 School of Information and Communication Engineering, Beijing University of Posts and Telecommunications (BUPT), Beijing 100876, China
2 Key Laboratory of Dynamic Cognitive System of Electromagnetic Spectrum Space, College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing 211106, China
3 School of Artificial Intelligence, Beijing University of Posts and Telecommunications (BUPT), Beijing 100876, China
4 School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China
* Correspondence: linlzp@nuaa.edu.cn

Abstract: Mobile edge computing (MEC) and device-to-device (D2D) communication can alleviate the resource constraints of mobile devices and reduce communication latency. In this paper, we construct a D2D-MEC framework and study the multi-user cooperative partial offloading and computing resource allocation. We maximize the number of devices under the maximum delay constraints of the application and the limited computing resources. In the considered system, each user can offload its tasks to an edge server and a nearby D2D device. We first formulate the optimization problem as an NP-hard problem and then decouple it into two subproblems. The convex optimization method is used to solve the first subproblem, and the second subproblem is defined as a Markov decision process (MDP). A deep reinforcement learning algorithm based on a deep Q network (DQN) is developed to maximize the amount of tasks that the system can compute. Extensive simulation results demonstrate the effectiveness and superiority of the proposed scheme.

Keywords: mobile edge computing; D2D communication; partial offloading; Q learning; deep Q-network

1. Introduction

In recent years, with the development of wireless networks and the popularity of smart mobile devices, mobile applications such as augmented reality (AR), virtual reality (VR), and facial recognition payment have grown exponentially [1,2]. These applications tend to be computation intensive and require low latency, but the battery capacities, computation resources, and storage capacities of mobile user equipment (UE) are very limited. As a result, most emerging applications may not be suitable for local execution on mobile devices [3,4]. To address this daunting challenge, the functions of the central network are increasingly moving towards the edge of the network [4].

Mobile edge computing (MEC) is regarded as a promising paradigm. This technique moves service platforms with computing, storage, and communication capabilities to the edge node (base station (BS)) nearest to the mobile devices [5-9]. MEC allows resource-constrained mobile terminals to migrate part or all of the complex applications to the edge cloud, becoming a low-latency, low-energy, and efficient solution [10-12]. However, the heterogeneous characteristics of BSs and MEC servers and the limited resources of communication and edge computing bring challenges to computational offloading methods [13].

Computing offloading is one of the key issues in MEC [14,15]. Its main task is to plan the offloading scheme of computing tasks and the allocation scheme of computing resources to reduce delays, save energy consumption, and improve computing resource...
utilization. There are two main task offloading strategies. In full offloading, each task can be completely offloaded to the resource device or run completely locally [2,16,17]. In partial offloading, each task can be partially offloaded to the resource device and partially left local [18–20].

Although the above studies have demonstrated the advantages of MEC computation offloading for improving wireless network computing performance, the limited computing resources of BSs are not always sufficient to support all mobile devices within their coverage. To solve this problem, some work studies offload computation to the neighboring devices through device-to-device (D2D) communication links [21–25]. The authors of [21] believe that each mobile user has its corresponding mobile peer; thus, the computing task can be partially offloaded to the edge cloud or its mobile peer. Successive convex approximation (SCA) and geometric programming (GP) are used to solve the energy minimization problem. Reference [22] solves the overall energy consumption problem of the system under the three-layer network architecture integrating cloud computing, MEC, and D2D communication through game theory. Reference [23] regards one near computing device with sufficient resources near each mobile user as a distributed computing node (DCN), and the mobile user can connect to the associated DCN through D2D. The problem of complete offloading of computing tasks between local, edge cloud, and DCN is studied relying on game theory, thereby reducing the delay and energy consumption of the system.

Although the above-mentioned research results have laid a solid foundation for utilizing the computing resources of idle devices in the MEC, the utilization efficiency of resources on idle devices has yet to be improved. By assuming that each mobile user is associated with a distributed computing node (DCN), Ref. [23] does not consider the DCN selection. Ref. [20] uses idle devices for D2D-assisted transmission and optimizes transmission scheduling, which does not use idle device resources for computing assistance. Although [25–27] optimizes the pairing between TD and RD, they stipulate that each device with an idle resource can only offer a D2D offloading service to one other device at most, and limits the pairing method between devices to one to one. Ref. [28] utilizes the resources of idle devices for computing assistance through D2D communication, and assumes that each idle device can accept multiple computing tasks. However, complete offloading is considered in this paper. Additionally, Ref. [28] assumes that when multiple computation tasks are offloaded to the same idle device, the tasks will share the computation resource equally.

Most of the works on MEC computing offload are aimed at reducing latency, saving energy, reducing energy efficiency, etc., and many state-of-the-art works only consider integrating multiple indicators for optimization [27,29]. However, the computing capability of the system can also be determined by the number of devices the system can serve. Because reduction in latency is often accompanied by the consumption of computing resources in the system, the number of devices supported by the system can be maximized subject to the constraint of maximum delay. In this way, the utilization efficiency of computing resources can be further improved without affecting user experience. As a result, the proposed scheme can cope with the continuous increase in the number of devices in the 6G network. In addition, traditional methods, such as dynamic programming, game theory, and GP, are often used to reduce the delay and energy consumption in full offloading, since such problems can be expressed as mixed-integer programming (MIP).

Compared with traditional optimization methods, reinforcement learning (RL) can deal with more complex MEC problems [30]. Reference [16] uses Q-learning and a deep Q network (DQN) to solve the binary offloading problem between the local and edge clouds in dynamic systems, and addresses the task assignment problem in a dynamic vehicular fog computing (VFC) environment. It simultaneously considers task priority, vehicle service availability, and computational resource sharing incentives, as well as using a soft actor–critic algorithm to maximize the utility of offloaded tasks.

In this paper, we consider adaptive user association, partial offloading, and resource allocation among multiple user devices and idle devices in a small area covered by a
single BS. Unlike traditional algorithms considering latency and energy consumption, we maximize the computing power of the entire system. An improved algorithm based on DQN is presented to solve the proposed problem. The main contributions of this paper are summarized as follows:

- We construct a D2D-MEC framework that combines D2D communications and MEC technology. The user equipment with limited computational capability can offload part of its computation-intensive tasks to the MEC server located in the BS and the idle equipment nearby, and the allocation of computing resources is the responsibility of the BS. In order to maximize the computing power of the whole system under the condition of limited computing resources, we propose an MEC framework including partial offloading, resource allocation, and user association under the maximum delay constraint of the application.

- We propose an optimization problem with constraints on both delay and computational resources, which is NP hard. By analyzing the internal structure of the optimization problem, it is decomposed into two sub-problems. We prove that the optimal solutions of the two sub-problems constitute the optimal solution of the original problem. The convex optimization is employed to solve the optimal solution of the first subproblem. The second subproblem is described as a Markov decision process (MDP) used to maximize the amount of tasks calculated by the system, in which offload decisions, resource allocation, and user association are determined simultaneously. A DQN-based modeless reinforcement learning algorithm is proposed to maximize the objective function.

- Extensive simulations demonstrate that the proposed algorithm outperforms traditional MEC schemes, Q-learning, DQN, and other conventional algorithms under different system parameters.

The rest of this paper is organized as follows. In Section 2, we present the system model, including a network model, a channel model, a computation model, and problem formulation. In Section 3, we decompose the original problem into two subproblems. In Section 4, we propose a reinforcement learning algorithm based on DQN to solve subproblem 2. In Section 5, we show the simulation results. Finally, we conclude this study in Section 6.

2. System Model

2.1. Network Model

As shown in Figure 1, the scenario we consider consists of a single BS equipped with an edge cloud server and mobile devices within the coverage area of the BS. A mobile device sends data to the BS through the cellular network and sends data to a nearby mobile device through D2D. Within the range of D2D communication, the mobile devices are divided into Task Devices (TDs), denoted as \( \mathcal{U} = \{ u_i | i = 1, 2, \ldots, U \} \), and D2D Resource Devices (D2D RDs), represented by \( k_i, i = 1, 2, \ldots, K \). Set \( \mathcal{K} = \{ k_i | i = 0, 1, 2, \ldots, K \} \) represents all resource devices, where \( k_0 \) is the BS. RDs can provide computing resources for tasks on TDs. The same as [31–33], the applications considered in this study are all oriented to data partitioning. The computing task on TD can be arbitrarily divided into three parts, and the computation is performed in parallel on the local, edge cloud, and a D2D RD at the same time.

We divide the system time into several time slots. The system state is constant in time slots, but changes between slots. Each slot BS allocates computing resources. The computing task on \( i \in \mathcal{U} \) is denoted as \( \phi_i = \{ Q_i, C_i, \tau_i, f_i \} \), where \( Q_i \) is the size of the task data, \( C_i \) indicates the number of CPU cycles per bit of data, representing the computational complexity of the application, \( \tau_i \) is the maximum delay, and \( f_i \) is the local computing capacity. Computing resources of \( k_i \in \mathcal{K} \) are denoted by \( F_{k_i}, i = 0, 1, 2, \ldots, K \).
Figure 1. Network model.

2.2. Channel Model

We assume that the wireless channel state remains constant when each computation task is transmitted between the TD and RD. The transmission rate between the TD $i$ and the BS is calculated by

$$R_{ik} = B_{ik} \log_2(1 + \frac{p_{ci}|h_{ik}|^2}{N_0}), \forall i \in U.$$  \hspace{1cm} (1)

The transmission rate between TD $i$ and D2D RD $j$ is calculated by

$$R_{ij} = B_{ij} \log_2(1 + \frac{p_{di}|h_{ij}|^2}{N_0}), \forall i \in U, \forall j \in K/k_0.$$  \hspace{1cm} (2)

where $h_{ij}$ is the channel power gain between TD $i$ and RD $j$; $B_{ij}$ is the bandwidth allocated to the cellular channel or D2D channel between TD $i$ and RD $j$; $p_{ci}$ is the transmission power of cellular from TD $i$ to BS; and $p_{di}$ is the transmission power of D2D from TD $i$ to a D2D RD. Since the two powers are limited by the maximum uplink power $p_{i}^{\text{max}}$ of TD, they are subject to constraints:

$$0 \leq p_{i} \leq p_{i}^{\text{max}},$$  \hspace{1cm} (3)

$$p_{i}^d = p_{i}^{\text{max}} - p_{i}, \forall i \in U.$$  \hspace{1cm} (4)

2.3. Computation Model

The computing task on TD is divided into three parts, which are computed on a local, edge cloud, and D2D RD, respectively. $x_{ij} \in \{0, 1\}, \forall i \in U, \forall j \in K/k_0$ is the user association between TD $i$ and RD $j$, $x_{ij} = 1$ indicates that TD $i$ offloads part of the computing task to D2D RD $j$, and otherwise, $x_{ij} = 0$. Since a TD selects, at most, one D2D RD for computational offloading, there are constraints: $\sum_{j=k_1}^{k} x_{ij} \leq 1, \forall i \in U$. Let $\alpha_i \in [0, 1], \beta_i \in [0, 1], i \in U$ denote the proportion of a computing task on TD $i$ that is offloaded to the edge cloud and D2D RD, respectively. Since the locally computed ratio should be non-negative, $\alpha_i$ and $\beta_i$ should satisfy the constraint: $0 \leq \alpha_i + \beta_i \leq 1$. Let $f_{ij}, \forall i \in U, \forall j \in K$ denote the computational resource allocated by RD $j$ to TD $i$. Since RDs have limited computing resources, there are constraints: $\sum_{i=U}^{U} x_{ij} f_{ij} \leq F_j, \forall j \in K$. 

• **Local Computing:** The local computation delay of the task on TD $i$ can be computed as

$$D_{lc}^i = \frac{(1 - \alpha_i - \beta_i)Q_iC_i}{f_i}. \quad (5)$$

• **Edge Computing:** The total latency of edge computing on TD $i$ consists of three parts: (1) time for uploading computing tasks $D_{et}^i$, (2) time for executing tasks on the MEC server $D_{ec}^i$, and (3) time for downloading computing results. Similar to [31,34], this study ignores the delay of sending results back to TDs from MEC server. This is because the size of the results is usually much smaller than the size of the transmitting data. Therefore, according to Equation (1), the delay of TD $i$ to complete edge cloud computing can be computed as

$$D_{ei}^i = D_{et}^i + D_{ec}^i = \frac{\alpha_iQ_iR_{ik_0}}{K} + \frac{\alpha_iQ_iC_i}{f_{ik_0}}. \quad (6)$$

• **D2D RDs Computing:** Similar to edge cloud computing, the delay of TD $i$ to complete D2D RD computing can be obtained by (1) D2D transmission delay $D_{dt}^i$, and (2) remote-execution delay $D_{dc}^i$

$$D_{di}^i = D_{dt}^i + D_{dc}^i = \sum_{j=k_1}^{k_0} x_{ij}(\frac{\beta_iQ_iR_{ij}}{f_{ij}} + \frac{\beta_iQ_iC_i}{f_{ij}}). \quad (7)$$

Therefore, according to Equations (5)–(7), the total delay for completing the task $\phi_i$ on TD $i$ is:

$$D_i = \max\{D_{lc}^i, D_{ei}^i, D_{di}^i\}. \quad (8)$$

### 2.4. Problem Formulation

In this paper, we improve the computing capability of the whole system, and the computing capability of the system is reflected by the number of devices supported by the system. Under the constraints of computing resources and maximum delay, the number of devices served by the system indicates the computing capability of the system [25].

We regard the number of TDs that can complete the computing task as the optimization target, and $o_{ui} = \begin{cases} 1 & \text{if } D_i \leq \tau_i, \\ 0 & \text{if } D_i > \tau_i \end{cases}$ is the completion of the computing task on TD $i$. $o_{ui} = 1$ indicates that the task is completed within the maximum delay; otherwise, $o_{ui} = 0$. The list of notations is given in abbreviations part. The optimization problem of this study can be formulated as:

$$\mathcal{P}1: \max_{\{x,a,b,f\}} \sum_{i=1}^{U} o_{ui}, \quad (9)$$

s.t. $x_{ij} \in \{0,1\}, \forall i \in U, \forall j \in K/k_0,$

$$\sum_{j=k_1}^{k_0} x_{ij} \leq 1, \forall i \in U,$$  \quad (10)

$$0 \leq a_i \leq 1, \forall i \in U,$$  \quad (11)

$$0 \leq \beta_i \leq 1, \forall i \in U,$$  \quad (12)

$$0 \leq a_i + \beta_i \leq 1, i \in U,$$  \quad (13)

$$\sum_{j=1}^{U} x_{ij}f_{ij} \leq F_j, \forall j \in K/k_0 \quad (14)$$

$$\sum_{j=1}^{U} f_{ik_0} \leq F_0, \quad (15)$$

$$\sum_{j=1}^{U} f_{ik_0} \leq F_0, \quad (16)$$
Theorem 1. \( \mathcal{P} 1 \) is an NP-hard problem.

**Proof.** See Appendix A. \( \square \)

Theorem 1 shows that \( \mathcal{P} 1 \) is a NP-hard problem, and the object function of (9) is non-convex, which is a difficult problem to solve. In order to solve \( \mathcal{P} 1 \), we first decompose and simplify the problem, and then solve it using a reinforcement learning method instead of a conventional optimization method.

3. Problem Decomposition

We maximize the number of TDs served by the system under the constraints of limited computing resources. The requirement for the completion of tasks on TDs is that the computing time is less than the maximum delay. According to \( D_i^{c_{max}} \) in Equation (6), the computing resource \( f_{ik} \) allocated by the edge server to TD \( i \) is proportional to the data size \( a_i \). Similarly, according to \( D_i^{D_{max}} \) in Equation (7), the computational resource \( f_{ij} \) allocated by D2D RD \( j \) to TD \( i \) is proportional to the data size \( \beta_i \). Based on the above analysis, reducing the value of \( \alpha_i + \beta_i \) can alleviate computing resource TD \( i \), which is occupied for the entire system. As a result, the system can have more computing resources to provide services for other TDs.

Based on the above analysis, in order to solve \( \mathcal{P} 1 \), we first determine the value of \( \alpha_i + \beta_i, \forall i \in \mathcal{U} \) and the resource allocation scheme \( f \), and then determine \( \alpha_i, \beta_i \) and user association \( x \). Finally, we prove that the optimal solution obtained in this way is the same as the optimal solution of \( \mathcal{P} 1 \). Define variables \( \gamma_i = \alpha_i + \beta_i, \alpha'_i = \frac{\alpha_i}{\alpha_i + \beta_i}, \beta'_i = \frac{\beta_i}{\alpha_i + \beta_i}, \forall i \in \mathcal{U} \). We have \( \alpha_i = \gamma_i \alpha'_i, \beta_i = \gamma_i \beta'_i \), and \( \beta'_i = 1 - \alpha'_i \). The variables in \( \mathcal{P} 1 \) change to \( \{ \gamma, \alpha', x, f \} \).

In \( \mathcal{P} 2 \), the values of \( \{ \alpha, x \} \) are fixed, and the optimal \( \{ \gamma, f \} \) are calculated. The goal of optimization is to minimize the demand for computing resources of the system. The mathematical formulation of the problem is expressed as

\[
\mathcal{P} 2 : \min_{\{ \gamma, f \}} \sum_{j=k}^{k_1} \sum_{i=1}^{u} x_{ij} f_{ij}, \quad (17)
\]

subject to

\[
0 \leq \gamma_i \leq 1, \forall i \in \mathcal{U}, \quad (18)
\]

\[
\frac{(1 - \gamma_i) c_i}{f_i} \leq \tau_i, \forall i \in \mathcal{U}, \quad (19)
\]

\[
\gamma_i a'_i Q_i \left( \frac{1}{R_{ik}} + \frac{c_i}{f_{ik}} \right) \leq \tau_i, \forall i \in \mathcal{U}, \quad (20)
\]

\[
\gamma_i (1 - \alpha'_i) Q_i \sum_{j=k}^{k_1} x_{ij} \left( \frac{1}{R_{ij}} + \frac{c_i}{f_{ij}} \right) \leq \tau_i, \forall i \in \mathcal{U}, \quad (21)
\]
where constraint (18) is set according to (14), and (19)–(21) represent constraints on local computing delay $D_l^c$, edge cloud computing delay $D_e^i$, and D2D computing delay $D_{D}^i$, respectively.

**Theorem 2.** The optimal solution of $P_2$ is given by $\{\gamma^*, f^*\}$:

$$\gamma^*_i = 1 - \frac{\tau f_i}{C_i}, \forall i \in U,$$

(22)

$$f_{ik_i^0}^* = \frac{C_i \gamma^*_i \alpha' Q_i R_{ik_i^0}}{\tau_i R_{ik_i^0} - \alpha' \gamma^*_i Q_i}, \forall i \in U,$$

(23)

$$f_{ij}^* = \frac{C_i \gamma^*_i \alpha' Q_i R_{ij}(1 - \alpha')}{{\tau_i R}_{ij} - \gamma^*_i (1 - \alpha' Q_i)}, \forall i \in U, \forall j \in K.$$

(24)

**Proof.** See Appendix B. $\Box$

It can be observed that $\gamma^*_i$ is a constant independent of $\{x, \alpha\}$, and $f_{ij}^*$ can be represented by variable $\alpha'$. Therefore, substituting the solution of $P_2$ into $P_1$ can obtain:

$P_3 : \max_{\{x, \alpha\}} \sum_{i=1}^{U} o'_{i_1},$

(25)

s.t. $x_{ij} \in \{0, 1\}, \forall i \in U, \forall j \in K/k_0,$

(26)

$$\sum_{j=k_1}^{k} x_{ij} \leq 1, \forall i \in U,$$

(27)

$$0 \leq \alpha'_i \leq 1, \forall i \in U,$$

(28)

$$\sum_{i=1}^{U} x_{ij} f_{ij} \leq F_j, \forall j \in K,$$

(29)

where $o'_{i_1}$ is obtained by

$$o'_{i_1} = \begin{cases} 1 & \text{if } \max\{D_{l,c}^i, D_{e}^i\} \leq \tau_i, \\ 0 & \text{if } \max\{D_{l,c}^i, D_{e}^i\} > \tau_i, \end{cases}$$

(30)

$$D_{l,c}^i = \frac{(1 - \gamma^*_i) C_i}{f_i} = \tau_i,$$

(31)

$$D_{e}^i = \gamma^*_i \alpha' Q_i (\frac{1}{R_{ik_i^0}} + \frac{C_i}{f_{ik_i^0}}),$$

(32)

$$D_{D}^i = \gamma^*_i (1 - \alpha') Q_i \sum_{j=k_1}^{k} x_{ij} (\frac{1}{R_{ij} f_{ij}}).$$

(33)

**Theorem 3.** The optimal solutions $\{\gamma^*, f^*\}, \{x^*, \alpha^{*}\}$ obtained by $P_2$ and $P_3$ are the optimal solutions of $P_1$.

**Proof.** See Appendix C. $\Box$

### 4. DQN-Based Computation Offloading

In order to solve problem $P_3$, we express it as a MDP, which can be solved using model-free reinforcement learning, and propose an improved reinforcement learning algorithm based on DQN. Compared with the conventional DQN, the improved reinforcement learning algorithm can enable the agent to learn a better solution and converge to solve the problems. In this paper, we improve the number of task users that the system can
serve by optimizing the offloading strategy and the allocation of computing resources in the system. Obviously, the optimal solution is unknown, so the agent can easily fall into a sub-optimal solution after finding a feasible allocation scheme and obtaining a positive reward. Therefore, in order to make the agent constantly search for a better solution, we compare and update the recorded action trajectory during the learning process, and then make random selection with a certain probability under the condition that the agent learns a lot of the recorded action trajectory, so that the agent has the opportunity to learn a better solution. In addition, we set the reward value given by the environment when learning the recorded action trajectory. This is different from using the Q-network to decide the action, which is used for fast convergence.

We call the proposed algorithm DQN-PTR, which integrates the priority action trajectory replay method into DQN. In this section, we first define the three key elements of the MDP problem, i.e., the state space, action space, and reward function. Then, we explain the detailed implementation of our proposed algorithm for reinforcement learning.

4.1. Three Key Elements for MDP

- **State Space:**
  At each time slot, the agent observes and collects all device information within the range of the BS. At step $t$, the state of the system consists of two parts: $s_t = \{F(t), \phi(t)\}$. Among them, $F = \{F_0(t), F_1(t), \ldots, F_K(t)\}$ represents the computing resources of all resource devices in the system, and $\phi(t) = \{\phi_{u_1}, \phi_{u_2}, \ldots, \phi_{u_U}\}$ where $\phi_i \in \{0, 1\}, i \in U$ represents the completion of the tasks on the task devices. We define $s_0$ as the system state observed by the BS at the beginning of the time slot; that is, $s_0 = \{F_0, F_1, \ldots, F_K, 0, 0, \ldots, 0\}$.

- **Action Space:**
  The action space consists of two parts: $A_t = \{x, \alpha'\}$, where $x = \{x_{u_1}, x_{u_2}, \ldots, x_{u_U}\}$, $x_i = \{x_{ik_1}, x_{ik_2}, \ldots, x_{ik_K}\}, i \in U$ represents the offload association between the TDs and the D2D RDs. $\alpha'$ is the task offload ratio of the currently assigned TDs. According to constraint (26) and (27), it is stipulated that action $a_t$ at step $t$ satisfies condition $\sum x(t) = 1$.

- **Reward Function:**
  The objective function of $P3$ is the sum of the TD devices that complete the calculation. Considering that the size of the computing tasks of each device is different, to ensure the fairness of the evaluation, the reward function is defined as the sum of the size of the computing tasks completed in the current time slot:

  $$ R(s_t, a_t) = \begin{cases} \sum_{i=1}^{U} o_{u_i} Q_i, & \text{if } \sum_{i=1}^{U} o_{u_i} > 0, \\ -1, & \text{else} \end{cases} $$

  (34)

In the reinforcement learning process, in each episode, as the environment performs the $t$ step action, the state and action space of the next step also change. Assume that the action selected by the agent at step $t$ is $a_t = \{x_{ij} = 1, \alpha'(t)\}$. $a_t$ is executed by the environment when $s_t$, $a_t$ satisfy the condition, given by

  $\phi_i = 0$,  
  $Q_i C_i \gamma_i \alpha'(t) \leq F_i(t)$,  
  $Q_i C_i(1 - \alpha'(t)) \leq F_0(t)$.  

(35)  
(36)  
(37)
With the successful execution of $a_t$, the remaining resources of the RDs and the TDs that have not allocated computing resources are reduced, and are given by

$$F_j(t + 1) = F_j(t) - Q_i C_i \gamma (t), \quad (38)$$
$$F_0(t + 1) = F_0(t) - Q_i C_i (1 - \gamma (t)), \quad (39)$$
$$A_{t+1} = A_t / x_i. \quad (40)$$

If any of the conditions (35)–(37) are not met, $a_t$ is infeasible, i.e., $s_{t+1} = s_t, A_{t+1} = A_t$.

4.2. Algorithm Design Based on DQN

The structure of the DQN-PTR algorithm proposed in this paper is shown in Figure 2, which mainly includes four parts: the environment, the replay buffer, the networks, and the trajectory record. The environment performs actions, computes rewards, and gives transitions to the state and the action space. The replay buffer stores the task offloading experiences, which are used to train the Q-Network. The network part includes two networks, which are used to predict the Q value and target the Q value, respectively. Network parameters are updated according to the differences between the two Q values. The recorded action trajectory is updated at the end of each training episode. If the result of the current episode is better than the recorded result, the action trajectory of the current episode is used to replace the originally recorded action trajectory.

![Figure 2. DQN-PTR based MEC system.](image)

We propose an DQN-PTR-based task offloading algorithm in Algorithm 1, and explain the main steps in detail, as follows:

1. Initialize the action-value function $Q(s, a)$ and the target action-value function $\hat{Q}(s, a)$ with parameters $\theta$ and $\theta'$, respectively. Initialize experiment replay buffer $D$ to an empty set of size $N$. 
2. Initialize $\epsilon = 0$, and specify that the growth rate of $\epsilon$ is $\epsilon_{\text{increment}} = 0.0001$ and grows to $\epsilon_{\text{max}} = 0.9999$. This parameter determines the probability of random selection when the agent selects an action, and the probability of random selection decreases with the update of the network parameters.
3. Initialize the optimal trajectory record $O = \emptyset$ and the maximum total return value $R = 0$.
4. In each episode, the agent in the BS collects the $s_0$. Initialize $R' = 0$ to calculate the total return value of this episode. $\mu \in [0, 1]$ is used to determine whether all actions in this episode are determined by the Q-Network or the optimal trajectory.
5. If the training episode is less than or equal to 100, $\epsilon$ is used to decide whether the choice of action is randomly selected or selected according to the maximum Q value.
6. When the training episode is greater than 100: $\mu > 0.9$, the selection of actions in this episode is the same as that in point number five; when $\mu \leq 0.9$, the actions are performed in accordance with the optimal trajectory $O$ in this episode. It should be noted that the reward settings in environment 1 and environment 2 are different. They are expressed as follows.

Environment 1:  
$$R(s_t, a_t) = \begin{cases} \sum_{i=1}^{U} o_{u_i} Q_i, & \text{if } \sum_{i=1}^{U} o_{u_i} > 0, \\ -1, & \text{else} \end{cases}$$

Environment 2:  
$$R(s_t, a_t) = \begin{cases} \sum_{i=1}^{U} o_{u_i} Q_i, & \text{if } \sum_{i=1}^{U} o_{u_i} > 0, \\ 0, & \text{else}. \end{cases}$$

7. When step $t$ ends, store $(s_t, a_t, r_t, s_{t+1})$ in the experience replay buffer; update the total reward value of the current cycle $R' = R' + r_t$; retrieve multiple records from the experience replay buffer to update Q-Network; increase the value of $\epsilon$.
8. At the end of each episode, compare the values of $R$ and $R'$, and compare the number of training steps in this period with the length of $O$ to determine whether to update the action trajectory record.

The advantages of the improved DQN-PTR includes:

- When adding a judgment item to the outer layer of the traditional DQN, the setting of the reward function becomes flexible.
- In addition to the traditional DQN replay memory $D$, a new cache space $O$ is added, which can be used to record the excellent action trajectory.
- Depending on the type of problem solved, the action space of the DQN-PTR can vary with the execution of each action.

Algorithm 1: DQN-PTR to solve $P3$

```
01: Initialize the Q-Network $Q$ with random weights $\theta$
02: Initialize the Target Q-Network $\hat{Q}$ with weights $\theta' = \theta$
03: Initialize replay memory $D$ to capacity $N$
04: Initialize $\epsilon = 0$, $\epsilon_{\text{increment}} = 0.0001$, $\epsilon_{\text{max}} = 0.9999$
05: Initialize optimal trajectory $O$ to empty and maximum total return $R = 0$
06: For episode = 1, $M$ do
07: Initialize sum reward $R' = 0$
08: Initialize state $s_0$, $\mu = \text{rand} \in [0,1]$
09: For each step $t$ do
10: If episode $\leq 100$ or $\mu > 0.9$ then
11: If $\text{rand} \in [0,1] \geq \epsilon$ then
12: Select a random action $a_t$
13: else
14: Set $a_t = \arg\max_a Q(s_t; a; \theta)$
15: end if
16: end if
17: Store $(s_t, a_t, r_t, s_{t+1})$ in $D$
18: Update $\epsilon$
19: Use multiple records from $D$ to update $Q$-Network
20: Retrieve multiple records from $D$ to update $\hat{Q}$-Network
21: Update $R' = R' + r_t$
22: end for
```
Algorithm 1: Cont.

16: Execute action $a_t$, observe next state $s_{t+1}$ and reward $r_t$ according to
17: environment 1
18: \textbf{else}
19: Set $a_t$ according to $O[t]$
20: \textbf{end if}
21: $R' = R' + r_t$
22: Store transition $\langle s_t, a_t, r_t, s_{t+1} \rangle$ in $D$
23: \textbf{If} episode terminates at step $t + 1$ \textbf{then}
24: \hspace{1em} Replace the trajectory in $O$ with $\{a_1, \ldots, a_t\}$
25: \hspace{1em} $R = R'$
26: \textbf{else if} $R' = R$ and $t < \text{len}(O)$ \textbf{then}
27: \hspace{1em} Replace the trajectory in $O$ with $\{a_1, \ldots, a_t\}$
28: \textbf{end if}
29: \textbf{break}
30: \textbf{end if}
31: \textbf{end if}
32: Sample random mini-batch of transitions $\langle s_i, a_j, r_j, s_{j+1} \rangle$ from $D$
33: \textbf{Set} $y_j = \begin{cases} r_j, & \text{if episode terminates at step } j + i, \\ r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta), & \text{otherwise}, \end{cases}$
34: \textbf{Perform a gradient descent step on } $(y_j - Q(s_j, a_j; \theta))^2$ \textbf{with respect to the network}
35: \hspace{1em} parameters $\theta$
36: \textbf{If} $\epsilon < \epsilon_{\text{max}}$ \textbf{then}
37: \hspace{1em} $\epsilon = \epsilon + \epsilon_{\text{increment}}$
38: \textbf{end if}
39: \textbf{Every C steps reset } $\hat{Q} = Q$
40: \textbf{end for}

5. Analysis of Simulation Results

In this section, we evaluate the performance of the computational offloading scheme proposed in this paper and the DQN-PTR algorithm through computer simulations. We first present the simulation parameters of the system. Then, we discuss the experimental results.

5.1. Simulation Setup

In the simulation, we assume the following scenario. We consider partial offloading between multiple devices in a small area covered by a single base station. The edge server on the base station side is equipped with an reinforcement learning agent which can make decisions about the offloading scheme in this area. In addition, the computing resources in the system are limited, and computing tasks cannot be completed locally. According to references [7,25,26,29,34,35], we set the simulation parameters to match our research scenario. We consider that the transmission power, channel bandwidth, and background noise of each device are 2 W, 10 MHz, and $-170$ dBm. The computing capacity of TDs and RDs are 24 Mcycles/s and 35 Mcycles/s, respectively. The data size and maximum latency of each task are 2.15 Mbits and 1 s, respectively. The number of CPU cycles is set to 20 Cycles/bit.

We compare the proposed algorithm with traditional MEC schemes, two benchmark algorithms, and reinforcement learning algorithms:

- **Full Local**: All TDs execute their tasks via local computing.
• **Local-cloud:** The computing tasks on TDs can be divided into two parts, which are computed on the local and edge cloud, respectively. In order to make full use of computing resources, all computing resources on TD $i$ are allocated to task $\phi_i$. If the local resources are insufficient, the computing resources will be supplemented by the edge cloud.

• **RBA:** The TD $i$ randomly selects a D2D RD device for computing offload and utilizes all the computing resources of the D2D RD device. If the computing resources of the two places are insufficient, the edge cloud will supplement the computing resources.

• **GBA:** The TD $i$ selects the D2D RD device with the largest remaining resources. Under the condition of making full use of local computing resources, the remaining computing tasks are evenly sent to the D2D RD device and edge cloud for computing.

• **Q-learning:** Q-learning is a basic reinforcement learning algorithm. Using Q-learning to solve $P3$, in simulation, the state space, action space, and reward function of Q-learning are all the same as those in DQN-PTR.

• **DQN:** DQN is an improved reinforcement learning algorithm based on Q-learning. In the simulation, the state space, action space, and reward function of DQN are all the same as those in DQN-PTR.

### 5.2. Simulation Result

In Figure 3, we show the number of supported (or unexecuted) TDs versus the total number of TDs in the system. The computing capacity of the MEC server is 50 Mcycles/s, and the number of D2D RDs is $\frac{1}{2}$ of the number of TDs. Since the scenarios we study mainly concern computationally intensive tasks, none of the tasks are computed locally. Relying on the computing resources of the local and MEC server, when the number of TDs reaches five, the upper limit of the system computing capacity can be reached. When considering D2D RDs in the system, the total number of computing tasks that the system can complete naturally increases as the number of TDs increases. In Figure 3, the DQN-PTR method proposed in this paper can achieve the best results. The DQN algorithm works satisfactorily when the number of devices is relatively small, but the results become worse as the number increases. However, the gap between Q-learning and DQN-PTR is always large. This is because as the number of devices increases, the number of actions and states in reinforcement learning also increases, which results in worse training results under the same training episodes. Additionally, when the number of TDs is greater than 10, the results obtained by the RBA and GBA algorithms are both smaller than the algorithm proposed in this paper.

In Figure 4, we show the number of supported (or unexecuted) TDs as the computing resources of the MEC server increases. The number of TDs and D2D RDs in the system is 30 and 15, respectively. It can be observed from Figure 4 that if the DQN-PTR algorithm is used for computing offload planning, the MEC server only needs to provide 100 Mcycles/s of computing resources to compute all tasks in the system. In addition, if the computing resources of D2D RDs in the system are utilized, even the relatively poor allocation algorithm (DQN, RBA and GBA) can save about 300 Mcycles/s of cloud computing resources compared with traditional local-cloud offloading.
In Figure 5, we present the number of supported (or unexecuted) TDs versus the number of D2D RDs. The computing power of the MEC server is fixed at 50 Mcycles/s and the number of TDs is 20, so the total data volume of all tasks in the system is 43.00 Mbits. As can be seen from Figure 5, our proposed DQN-PTR algorithm requires the fewest D2D RD devices. DQN, GBA, and Q-learning algorithms require 15 D2D RD devices to complete all tasks, while RBA requires 20 D2D RD devices.
Figure 5. (a) The number of supported TDs versus the number of RDs. (b) The number of unexecuted TDs versus the number of RDs.

Figure 6 shows the learning curves of Q-learning, DQN, and DQN-PTR. The number of TDs and D2D RDs is 20 and 10, respectively. In order to ensure fairness, the three algorithms have the same parameters, i.e., the maximum number of steps allowed per episode is 50. The $\epsilon$-greedy value increases from 0 to 0.95 with a growth rate of 0.0001. Learning rate $\gamma$, replay memory size, and mini-batch size are 0.0001, 10,000, and 200, respectively. Reference [36] for an analysis of our curves, we found that DQN-PTR performed better than DQN and Q-learning. Although Q-learning has the fastest learning speed, our proposed algorithm is more stable than DQN in terms of the fluctuation of the curve, and obtains the highest average return value.

6. Conclusions

This paper has proposed an integrated framework for multi-user partial offloading and resource allocation, combining MEC and D2D technologies. Under this framework, we can make the decision of computational offloading and the resource allocation of MEC and idle devices. We have also designed a convex optimization method to simplify the
problem. Finally, we have derived a DQN-based reinforcement learning algorithm to solve the problem. Simulation results have shown that the proposed scheme has better performance than other benchmark schemes under different system parameters.

**Author Contributions:** Conceptualization, X.G. and T.L.; methodology, X.G.; software, X.G.; validation, X.G., T.L. and Z.L.; formal analysis, X.G.; investigation, T.L.; writing—original draft preparation, X.G.; writing—review and editing, Z.L., J.Z. and P.H. 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 62271068, 61827801, and in part by the Basic Scientific Research Project under Grant NS2022046.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

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

**Abbreviations**

The following abbreviations are used in this manuscript:

- MEC: Mobile edge computing
- D2D: Device-to-device
- MDP: Markov decision process
- DQN: Deep Q network
- AR: Augmented reality
- VR: Virtual reality
- UE: User equipment
- BS: Base station
- SCA: Successive convex approximation
- GP: Geometric programming
- DCN: Distributed computing node
- MIP: Mixed integer programming
- RL: Reinforcement learning
- VFC: Vehicular fog computing
- TD: Task device
- RD: Resource device
- D2D RD: D2D Resource device

**Notations**

- $u_i$: The index of the TD $i$
- $k_i$: The index of the RD $i$, where $k_0$ represents the BS, and the others represent D2D RD
- $U$: The number of TDs
- $K$: The number of D2D RDs
- $\mathcal{U}$: The set of all TDs
- $\mathcal{K}$: The set of all RDs
- $\phi_i$: The index of the computing task on $i \in \mathcal{U}$
- $Q_i$: The data size of the task $\phi_i$
- $C_i$: CPU cycles per bit required for task $\phi_i$
- $\tau_i$: The maximum delay of the task $\phi_i$
- $f_i$: The local computing capacity of the TD $i$
- $F_i$: The computing resource of the RD $i$
- $R_{ij}$: The transmission rate between the TD $i$ and the RD $j$
- $B_{ij}$: The bandwidth allocated to the channel between TD $i$ and RD $j$
- $p_{ij}$: The cellular transmission power from TD $i$ to BS
The transmission power of D2D from TD $i$ to a D2D RD $p_d$

The maximum uplink power of TD $i$ $p_{\text{max}}$

User association between TD $i$ and RD $j$ $x_{ij}$

The proportion of a computing task on TD $i$ that is offloaded to the BS $a_i$

The proportion of a computing task on TD $i$ that is offloaded to D2D RD $b_i$

The local computation delay of the task on TD $i$ $D_{i,c}$

The delay of TD $i$ to complete edge cloud computing $D_{e,i}$

Cellular transmission delay of TD $i$ $D_{e,c,i}$

The computation delay of task $\phi_i$ at BS $D_{\phi,i}$

The delay of TD $i$ to complete D2D RD computing $D_{D,i}$

D2D transmission delay of TD $i$ $D_{D,c,i}$

The computation delay of task $\phi_i$ at D2D RD $D_{\phi,D,i}$

The total delay for completing the task $\phi_i$ on TD $i$ $D_{i}$

The completion of the computing task on TD $i$ $o_{ui}$

**Appendix A**

Considering that when the MEC server and all TDs do not provide computing resources, and the transfer time is fast enough. In this case, the number of computing tasks is much more than the number of D2D RDs, and satisfies:

$$\phi_i = \phi_{\cup}, \forall i \in \mathcal{U},$$

$$F_j = F_{K}, \forall j \in \mathcal{K}/k_0.$$ 

Therefore, the maximum number of tasks supported by a single D2D RD is $F_{K}/Q_i$. In $P1$, the problem of maximizing the number of tasks can be rewritten as

$$P_A : \max x \sum_{j=k_1}^{k_k} (\sum_{i=ui}^{u_U} x_{ij}),$$

s.t. $x_{ij} \in \{0, 1\}, \forall i \in \mathcal{U}, \forall j \in \mathcal{K}/k_0$,

$$\sum_{j=k_1}^{k_k} x_{ij} = 1, \forall i \in \mathcal{U},$$

$$\sum_{i=ui}^{u_U} x_{ij} \leq \frac{F_{K}}{Q_i C_i}, \forall j \in \mathcal{K}/k_0.$$ 

It is difficult to obtain the optimal solution of the problem $P_A$. For the fact that $x$ is binary, then the feasible set is not convex, $P_A$ is non-convex integer programs. According to the statement in [17,37,38], $P_A$ is NP-hard. From [39], $P1$ is also NP-hard because $P1$ is a special case of $P_A$.

**Appendix B**

According to (10) and (11), $\sum_{j=k_1}^{k_k} x_{ij}$ is equal to 0 or 1. Therefore, in order to prove Theorem 1, the problem is classified according to the value of user association $x$.

- When $\sum_{j=k_1}^{k_k} x_{ij} = 0$, the computing task of TD $i$ is only computed at the local and edge cloud, thus $b'_i = 0, f_{ij} = 0, \forall j \in \mathcal{K}/k_0$ and $a'_i = 1$. In this case, constraint (21) is obviously satisfied. We then have

$$\gamma_i \geq 1 - \frac{\tau_i f_i}{C_i}.$$  

(A1)

From (20), we can obtain

$$f_{\tilde{k}0} \geq \frac{C_i a'_i \gamma_i Q_{\tilde{k}0}}{\tau_i R_{\tilde{k}0} - a'_i \gamma_i Q_i}.$$  

(A2)
Take the derivative of the right hand side of the above inequality, we have
\[
\frac{d}{d\gamma_i} \left( \frac{C_i\alpha_i'\gamma_i Q_i R_i k_i}{\tau_i R_i k_i - \alpha_i'\gamma_i Q_i} \right) = \frac{C_i\alpha_i'Q_iR_i k_i \tau_i}{(\tau_i R_i k_i - \alpha_i'\gamma_i Q_i)^2} > 0. \tag{A3}
\]
To sum up, when the equal signs of (A1) and (A2) hold, \( f_{ik_0} \) can take the minimum value, that is, (22) and (23) are proved.

- When \( \sum_{i=k}^{k_{\text{th}}} x_{ij} = 1 \), it means that TD \( i \) offloads the computing task to a certain D2D RD, we define the D2D RD as D2D RD \( j \). According to (21), we have
  \[
  f_{ij} = \frac{C_i\gamma_i Q_i R_i (1 - \alpha_i')}{\tau_i R_i - \gamma_i (1 - \alpha_i') Q_i}. \tag{A5}
  \]
  Similar to (A2), when the equal signs of (A1) and (A5) hold, \( f_{ij} \) takes the minimum value, that is, (24) can be proved.

- When \( \sum_{i=k}^{k_{\text{th}}} x_{ij} = 0, \alpha_i' = 1 \) is substituted into (24) and the result is \( f_{ij}^* = 0 \). Therefore, Theorem 1 is proved in both cases.

**Appendix C**

In order to prove that the optimal solutions of \( \mathcal{P}2 \) and \( \mathcal{P}3 \) are the same as the optimal solutions of \( \mathcal{P}1 \), it is necessary to prove that the maximum values of the objective functions calculated by the optimal solutions obtained by \( \mathcal{P}3 \) and \( \mathcal{P}1 \) are the same. To this end, we first denote the optimal solution set of \( \mathcal{P}2 \) and \( \mathcal{P}3 \) as: \( \{\gamma^*, f^*, x^*, \alpha'^*\} \). The maximum value of the objective function of \( \mathcal{P}3 \) is \( N_3 \). The optimal solution set of \( \mathcal{P}1 \) is set as:
\[
\{x_i^*, \alpha_i^*, \beta_i^*, f_i^*\}, \text{ where } \gamma_i^* = \alpha_i^* + \beta_i^* \alpha_i'^* = \frac{\alpha_i^*}{\gamma_i^*}, \beta_i^* = \gamma_i^* - \alpha_i'^*, \text{ and the maximum value of the objective function is } N_1.
\]
From (18), (26)–(28), it is easy to know that \( \gamma^*, x^*, \alpha'^* \) satisfy (10)–(14), and from (29), it can be deduced that \( f^* \) calculated by \( \alpha'^* \) satisfies (15). To sum up, \( \{\gamma^*, f^*, x^*, \alpha'^*\} \) are feasible solutions of \( \mathcal{P}1 \), from which \( N_3 \leq N_1 \) can be deduced.

Similarly, it can be proved that \( \{x_i^*, \alpha_i^*, \beta_i^*, f_i^*\} \) satisfy the constraints (18), (26)–(29). Assuming that the conditions (19)–(21) are not satisfied, it means that there are offloading schemes \( \{x_i^*, \alpha_i^*, \beta_i^*, f_i^*\} \) of TDs, \( i \in \mathcal{U} \) that do not satisfy the constraints (19)–(21), that is, \( D_i > \tau_i, o_{ij} = 0, f_{ij}^* \leq 0, 0 \leq \alpha_i^* + \beta_i^* \leq 1 \). This means that the TD \( i \) and all computing resources \( f_{ij}^* = \{f_{ij}^*\}, \forall j \in \mathcal{K} \) allocated to TD \( i \) does not affect the value of the objective function. Group all TDs that do not satisfy the constraints into set \( \mathcal{M} = \{m_i | i = 1, 2, \ldots, U - M\} \). Set \( \mathcal{N} = \mathcal{U} / \mathcal{M} = \{n_j | i = 1, 2, \ldots, U - M\} \) and redefine the computing resource in the system as \( F_j = F_j - \sum_{i=1}^{M} f_{im,j}^* \). Re-resource allocation is performed on TDs set \( \mathcal{N} \) and network resource state \( F_i \) through \( \mathcal{P}1 \), and the obtained resource allocation scheme is equivalent to \( \{x_i^*, \alpha_i^*, \beta_i^*, f_i^*\} \) and both satisfy the constraints of \( \mathcal{P}2 \) and \( \mathcal{P}3 \). So the maximum value of the objective function is still \( N_1 \) and the solution is a feasible solution of \( \mathcal{P}3 \). The maximum objective function obtained by solving the same problem through \( \mathcal{P}2, \mathcal{P}3 \) is defined as \( N'_3 \), so \( N_1 \leq N'_3 \). In addition, since the number of TDs and the computing resources in the network are both reduced, the obtained maximum objective function value \( N'_3 \leq N_3 \), so \( N_1 \leq N'_3 \leq N_3 \).

To sum up, \( N_1 \leq N'_3 \leq N_3 \) and \( N_3 \leq N_1 \), so there can only be \( N_3 = N_1 \). Theorem 2 is proved.
References

1. Mangiante, S.; Klas, G.; Navon, A.; GuanHua, Z.; Ran, J.; Silva, M.D. Vr is on the edge: How to deliver 360 videos in mobile networks. In Proceedings of the Workshop on Virtual Reality and Augmented Reality Network, New York, NY, USA, 8–11 August 2017; pp. 30–35.

2. Yang, Z.; Liu, Y.; Chen, Y.; Tyson, G. Deep reinforcement learning in cache-aided MEC networks. In Proceedings of the IEEE 2019—2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6.

3. Dinh, T.Q.; Tang, J.; La, Q.D.; Quek, T.Q.S. Offloading in Mobile Edge Computing: Task Allocation and Computational Frequency Scaling. IEEE Trans. Commun. 2017, 65, 3571–3584.

4. Feng, C.; Han, P.; Zhang, X.; Yang, B.; Liu, Y.; Guo, L. Computation offloading in mobile edge computing networks: A survey. J. Netw. Comput. Appl. 2022, 202, 103366–103381. [CrossRef]

5. Elgendy, I.A.; Zhang, W.Z.; Zeng, Y.; He, H.; Tian, Y.-C.; Yang, Y. Efficient and secure multi-user multi-task computation offloading for mobile-edge computing in mobile IoT networks. IEEE Trans. Netw. Serv. Manag. 2020, 17, 2410–2422. [CrossRef]

6. Khayyat, M.; Elgendy, I.A.; Muthanna, A.; Alshahran, A.S.; Alharbi, S.; Koucheryavy, A. Advanced deep learning-based computational offloading for multilevel vehicular edge-cloud computing networks. IEEE Access 2020, 8, 137052–137062. [CrossRef]

7. Zhang, W.Z.; Elgendy, I.A.; Hammad, M.; Iliyasu, A.M.; Du, X.; Guizani, M.; El-Latif, A.A.A. Secure and optimized load balancing for multi-tier IoT and edge-cloud computing systems. IEEE Internet Things J. 2020, 8, 8119–8132. [CrossRef]

8. Taleb, T.; Sandanis, K.; Mada, B.; Flinck, H.; Dutta, S.; Sabella, D. On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration. IEEE Commun. Surv. Tutor. 2017, 19, 1657–1681. [CrossRef]

9. Shakarami, A.; Shahidinejad, A.; Ghobaei-Arani, M. An autonomous computation offloading strategy in Mobile Edge Computing: A deep learning-based hybrid approach. J. Netw. Comput. Appl. 2021, 178, 102974–102992. [CrossRef]

10. Mach, P.; Becvar Z. Mobile edge computing: A survey on architecture and computation offloading. IEEE Commun. Surv. Tutor. 2017, 19, 1628–1656. [CrossRef]

11. Hu, Z.; Niu, J.; Ren, T.; Dai, B.; Li, Q.; Xu, M.; Das, S.K. An efficient online computation offloading approach for large-scale mobile edge computing via deep reinforcement learning. IEEE Trans. Serv. Comput. 2021, 15, 669–683. [CrossRef]

12. Kumari, P.; Mishra, R.; Gupta, H.P.; Dutta, T.; Das, S.K. An energy efficient smart metering system using edge computing in LoRa network. IEEE Trans. Sustain. Comput. 2021, 1–13. [CrossRef]

13. Othman, M.; Madani, S.A.; Khan, S.U. A survey of mobile cloud computing application models. IEEE Commun. Surv. Tutor. 2013, 16, 393–413.

14. Wu H. Multi-objective decision-making for mobile cloud offloading: A survey. IEEE Access 2018, 6, 3962–3976. [CrossRef]

15. Chalaemwongwan, N.; Kurutach W. Mobile cloud computing: A survey and propose solution framework. In Proceedings of the 2016 13th International Conference on Electrical Engineering/Electronics,Computer,Telecommunications and Information Technology (ECTI-CON), Chiang Mai, Thailand, 28 June–1 July 2016; pp. 1–4.

16. Li, J.; Gao, H.; Lv, T.; Lu, Y. Deep reinforcement learning based computation offloading and resource allocation for MEC. In Proceedings of the 2018 IEEE Wireless Communications and networking conference (WCNC), Barcelona, Spain, 15–18 April 2018; pp. 1–6.

17. Elgendy, I.A.; Zhang, W.Z.; He, H.; Gupta, B.B.; El-Latif, A.A.A. Joint computation offloading and task caching for multi-user and multi-task MEC systems: Reinforcement learning-based algorithms. Wirel. Netw. 2021, 27, 2023–2038. [CrossRef]

18. Sahni, Y.; Cao, J.; Yang, L.; Ji, Y. Multi-hop multi-task partial computation offloading in collaborative edge computing. IEEE Trans. Parallel Distrib. Syst. 2020, 32, 1133–1145. [CrossRef]

19. Chouhan, S. Energy optimal partial computation offloading framework for mobile devices in multi-access edge computing. In Proceedings of the 2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 19–21 September 2019; pp. 1–6.

20. Peng, J.; Qiu, H.; Cai, J.; Xu, W.; Wang, J. D2D-assisted multi-user cooperative partial offloading, transmission scheduling and computing allocating for MEC. IEEE Trans. Wirel. Commun. 2021, 20, 4858–4873. [CrossRef]

21. Ti, N.T.; Le, L.B. Computation offloading leveraging computing resources from edge cloud and mobile peers. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–6.

22. Wang, C.; Qin, J.; Yang, X.; Wen, W. Energy-efficient offloading policy in D2D underlay communication integrated with MEC service. In Proceedings of the 3rd International Conference on High Performance Compilation, Computing and Communications. Xi’an, China, 8–10 March 2019; pp. 159–164.

23. Hu, G.; Jia, Y.; Chen Z. Multi-user computation offloading with d2d for mobile edge computing. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6.

24. Zhou, Z.; Dong, M.; Ota, K.; Wang, G.; Yang, L.T. Energy-efficient resource allocation for D2D communications underlaying cloud-RAN-based LTE-A networks. IEEE Internet Things J. 2015, 3, 428–438. [CrossRef]

25. He, Y.; Ren, J.; Yu, G.; Cai, Y. D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks. IEEE Trans. Wirel. Commun. 2019, 18, 1750–1763. [CrossRef]

26. Chai, R.; Lin, J.; Chen, M.; Chen, Q. Task Execution Cost Minimization-Based Joint Computation Offloading and Resource Allocation for Cellular D2D MEC Systems. IEEE Syst. J. 2019, 13, 4110–4121. [CrossRef]

27. Hamdi, M.; Hamed, A.B.; Yuan, D.; Zaid, M. Energy-Efficient Joint Task Assignment and Power Control in Energy-Harvesting D2D Offloading Communications. IEEE Internet Things J. 2022, 9, 6018–6031. [CrossRef]
28. Fang, T.; Yuan, F.; Ao, L.; Chen, J. Joint Task Offloading, D2D Pairing, and Resource Allocation in Device-Enhanced MEC: A Potential Game Approach. *IEEE Internet Things J.* 2022, 9, 3226–3237. 10.1109/JIOT.2021.3097754. [CrossRef]

29. 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, 14, 1–14. [CrossRef]

30. Shakarami, A.; Ghobaei-Arani, M.; Shahidinejad, A. A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective. *Comput. Netw.* 2020, 182, 107496–107519. [CrossRef]

31. Qin, M.; Cheng, N.; Jing, Z.; Yang, T.; Xu, W.; Yang, Q.; Rao, R.R. Service-oriented energy-latency tradeoff for IoT task partial offloading in MEC-enhanced multi-RAT networks. *IEEE Internet Things J.* 2020, 8, 1896–1907. [CrossRef]

32. Guo, M.; Wang, W.; Huang, X.; Chen, Y.; Zhang, L.; Chen, L. Lyapunov-based Partial Computation Offloading for Multiple Mobile Devices Enabled by Harvested Energy in MEC. *IEEE Internet Things J.* 2021, 9, 9025–9035. [CrossRef]

33. Truong, T.P.; Nguyen, T.V.; Noh, W.; Cho, S. Partial computation offloading in NOMA-assisted mobile-edge computing systems using deep reinforcement learning. *IEEE Internet Things J.* 2021, 8, 13196–13208. [CrossRef]

34. Shi, J.; Du, J.; Wang, J.; Yuan, J. Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning. *IEEE Trans. Veh. Technol.* 2020, 69, 16067–16081. [CrossRef]

35. Saleem, U.; Liu, Y.; Jangsher, S.; Tao, X.; Li, Y. Latency Minimization for D2D-Enabled Partial Computation Offloading in Mobile Edge Computing. *IEEE Trans. Veh. Technol.* 2020, 69, 4472–4486. [CrossRef]

36. Ohnishi, S.; Uchibe, E.; Yamaguchi, Y.; Nakanishi, K.; Yasui, Y.; Ishii, S. Constrained deep q-learning gradually approaching ordinary q-learning. *Front. Neurorobot.* 2019, 13, 103–116. [CrossRef]

37. Fooladivanda, D.; Rosenberg, C. Joint Resource Allocation and User Association for Heterogeneous Wireless Cellular Networks. *IEEE Trans. Wirel. Commun.* 2013, 12, 248–257. [CrossRef]

38. Bu, T.; Li, L.; Ramjee, R. Generalized Proportional Fair Scheduling in Third Generation Wireless Data Networks. In *Proceedings of the IEEE INFOCOM 2006—25th IEEE International Conference on Computer Communications*, Barcelona, Spain, 10 April 2007; pp. 1–12.

39. Garey, M.R.; Johnson, D.S. “Strong” np-completeness results: Motivation, examples, and implications. *J. ACM* 1978, 25, 499–508. [CrossRef]