electronics

MDPI

Article

DRL-Based Computation Offloading and Resource Allocation in Green MEC-Enabled Maritime-IoT Networks

Ze Wei, Rongxi He *, Yunuo Li and Chengzhi Song

College of Information Science and Technology, Dalian Maritime University, Dalian 116026, China; wllt@dlmu.edu.cn (Z.W.); liyunuo@dlmu.edu.cn (Y.L.); scz@dlmu.edu.cn (C.S.)
* Correspondence: hrx@dlmu.edu.cn

Abstract: The maritime Internet of Things (MIoT), a maritime version of the Internet of Things (IoT), is envisioned as a promising solution that can provide ubiquitous connectivity over land and sea. Due to the rapid development of maritime activities and the maritime economy, there is a growing demand for computing-intensive and latency-sensitive maritime applications requiring various energy consumption, communication, and computation resources, posing a significant challenge to MIoT devices due to their limited computational ability and battery capacity. Mobile Edge Computing (MEC), which can handle computation tasks at the network's edge more efficiently and with less latency, is emerging as a paradigm for fulfilling the ever-increasing demands of MIoT applications. However, the exponential increase in the number of MIoT devices has increased the system's energy consumption, resulting in increased greenhouse gas emissions and a negative impact on the environment. As a result, it is vital for MIoT networks to take traditional energy usage minimization into account. The integration of renewable energy-harvesting capabilities into base stations or MIoT devices possesses the potential to reduce grid energy consumption and carbon emissions. However, making an effective decision regarding task offloading and resource allocation is crucial for maximizing the utilization of the system's potential resources and minimizing carbon emissions. In this paper, we first propose a green MEC-enabled maritime IoT network architecture to flexibly provide computing-intensive and latency-sensitive applications for MIoT users. Based on the architecture, we formulate the joint task offloading and resource allocation problem by optimizing the total system execution efficiency (including the total size of completed tasks, task execution latency, and the system's carbon emissions) and then propose a deep-deterministic-policy-gradient-based joint optimization strategy to solve the problem, eventually obtaining an effective resolution through continuous action space learning in the changing environment. Finally, simulation results confirm that our proposal can yield good performance in system execution efficiency compared to other benchmarks; that is, it can significantly reduce the system's carbon emissions and tasks' delay and improve the total size of completed tasks.

Keywords: maritime Internet of things (MIoT); mobile edge computing (MEC); computation offloading; carbon emissions; renewable energy; deep deterministic policy gradient (DDPG)

Citation: Wei, Z.; He, R.; Li, Y.; Song, C. DRL-Based Computation Offloading and Resource Allocation in Green MEC-Enabled Maritime-IoT Networks. *Electronics* **2023**, *12*, 4967. https://doi.org/10.3390/electronics12244967

Academic Editors: Fernando Reinaldo Ribeiro and José Metrôlho

Received: 6 November 2023
Revised: 9 December 2023
Accepted: 10 December 2023
Published: 11 December 2023

1. Introduction

The ocean not only contains numerous biological and mineral resources but also plays a significant role in the study of biodiversity, the exploration of global climate change, the development of marine resources, and ocean transportation [1,2]. In recent years, the ocean has become a focal point of great concern for governments, academia, and industry, prompting a significant acceleration in their efforts in ocean exploration and advance maritime communication. The Internet of Things (IoT) allows a wide range of smart devices, from simple smart home devices to complex industrial machinery, to gather extensive data and communicate with each other and with other internet-connected devices. This seamless data exchange enables these devices to carry out a variety of tasks autonomously [3]. The

success of the IoTs in the mainland has prompted extensive exploration into how to develop a maritime version of the Maritime Internet of Things (MIoT) [4,5]. MIoT is able to bring the advantages of the IoT to the maritime sector and provide ubiquitous internet services for marine environmental monitoring and emerging maritime-related applications, such as marine pollution monitoring, tidal recording, ocean current measuring, assisted navigation, maritime search and rescue (SAR) operations, vessel traffic management, and so on.

Similar to IoT devices (IoTD), MIoT devices (MIoTD), such as various sensors, smart buoys, unmanned surface vessels (USV), and floats, have stringent computation and energy resource limitations. To reduce the burden of powering and computing for task computation and reduce task processing latency at terminals, Mobile Edge Computing (MEC) is gradually replacing the cloud computing paradigm, envisioned as a promising paradigm, by putting lightweight servers in closer proximity to the terminals [6]. MEC architecture can be employed in a variety of IoT application scenarios, like 6G communication, virtual reality, the Internet of Vehicles (IoV), smart cities, smart factories, and more. As one of the core techniques of the MEC, task offloading has received great attention recently. Most of the work has been conducted to determine how computation offloading can be optimized, that is, ensuring rapid and reliable wireless transmission while also providing a reasonable amount of communication and computation resources to meet task latency requirements [7]. Generally speaking, binary offloading approaches are usually employed for simple, indivisible, or highly integrated tasks. This means that tasks can be either processed locally or entirely sent to the MEC servers [8]. The authors in [9,10] studied the binary offloading problem with the goal of minimizing energy consumption, and they found that utilizing MEC servers for certain computing tasks can be more effective in certain situations.

Except that the implementation of MIoT is impeded by computing resource constraints, the shift in dominant traffic in wireless networks is a significant driving factor for the progression toward 6G. Since there is a finite amount of the spectrum available in wireless networks, advanced channel coding and modulation techniques are required to reduce interference, which normally infers more power consumption, not only from the transceiver but also from the overall radio access network [11]. It is anticipated that wireless networks will remain a major contributor to the global carbon footprint, which is predicted to double in the next decade [12]. Hence, a key challenge in deploying the MIoT is minimizing its traditional energy consumption to lower its carbon footprint and environmental negative impact. Furthermore, due to the numerous MIoTDs involved in the MIoT, such as underwater sensors and smart buoys on the sea, replacing or recharging the batteries of these devices is not convenient, not to mention the relevant cost and effectiveness [13]. Therefore, adopting more energy-efficient, sustainable, and eco-friendly wireless communication systems is becoming increasingly necessary. The integration of MIoT and green communication technology is anticipated to bring a multitude of benefits by optimizing joint computing and resource allocation to reduce their traditional energy consumption and carbon footprint. For instance, by implementing energy-efficient protocols and equipment in MIoT, its power consumption can be cut back, while the deployment of renewable energy sources can further help to balance the carbon emissions associated with the network's operation.

As an achievable way to cut down on traditional energy usage and carbon emissions, harnessing renewable energy [14,15] has attracted much attention in recent years. By incorporating energy harvest (EH) modules into energy storage devices, base stations (BSs), or other equipment, renewable energy, such as solar energy, wind energy, and tidal energy, can be exploited and then converted into electric power. This helps reduce carbon emissions and enables self-maintenance circuits [16]. A few recent works have considered the integration of EH and MEC and the optimization of task offloading [8,17,18]. These prior studies have primarily emphasized enhancing computing efficiency and minimizing energy consumption in IoTDs to enhance performance in the "energy-saving approach". However, the aspect of reducing carbon emissions in the "exploiting renewable energy approach" has not received adequate attention in these works. With the increasing popularity of

the internet and broadband communication systems, it is essential to consider reducing these systems' traditional energy consumption to decrease their carbon footprints and keep their environmental impact to a minimum. Furthermore, offloading decisions can be more flexible. However, current classical network optimization algorithms still rely on complex heuristic adjustments to find a sufficient solution, resulting in exponential computational time and limited effectiveness for larger networks. Fortunately, deep reinforcement learning (DRL) techniques can replace the laborious process of traditional optimization algorithms while reducing computational complexity [19,20]. Recently, more and more attention has been given to the DQN-based strategies [21,22] applied to the MEC offloading issue. Nevertheless, value-based methods are not sufficient to tackle the optimization challenges of dynamic computation offloading and resource allocation with a continuous action space. This is due to their inability to extract temporal information from task data as they are restricted to global discrimination. To overcome the inefficiency and high variance associated with assessing a policy using policy gradient methods, the authors in [23,24] proposed joint optimization methods based on the DDPG structure that involves the simultaneous continuous-time adaptation of both actor and critic neural networks. However, the work [24] solely focused on minimizing the system's energy consumption, ignoring the fact that energy savings in a hybrid energy system do not imply lower carbon emissions.

Motivated by the above, in our previous work [25], we adapted the deep deterministic policy gradient (DDPG) for joint optimization of computation offloading and resource allocation schemes, aiming at maximizing the total system execution efficiency (including the total size of completed tasks, task execution latency, and the system's carbon emissions) under a hybrid energy supply. As an extension of the earlier work in [25], in this article, we consider in more detail the constraints of task latency, computational resource capacity, and transmission power to formulate the maximizing system execution efficiency problem in the green MEC-enabled MIoT network and adapt the DDPG to effectively optimize the computation offloading and resource allocation scheme, aiming to significantly reduce the system's carbon emissions and tasks' delay and improve the total size of completed tasks. The difference between this paper and our previous work is listed in Section 2. The primary findings of this study are outlined below:

- Network architecture and problem formulation: We first propose a green MEC-enabled MIoT network architecture with multiple MIoTDs and multiple BSs and then formulate the problem of joint task offloading and resource allocation in the green MEC-enabled MIoT network as a stochastic optimization problem to maximize system execution efficiency, which consists of system execution cost (including system task latency and carbon emissions) and the size of completed tasks. Our objective function is more general than those proposed by many related studies in that we include the system's carbon emissions due to fossil energy usage, the size of completed tasks, and their latency.
- Algorithm design: We propose a DDPG-based carbon-aware task offloading and resource allocation algorithm (DCTORA). We jointly consider maximizing the size of completed tasks and minimizing the system's carbon emissions and the delay of tasks. DCTORA is a model-free DRL method that can efficiently handle continuous action space. Moreover, it can determine the joint optimization scheme for each task in an unpredictable environment with stochastic tasks and renewable energy sources.
- Experimental stimulation: The performance of our proposal is evaluated by extensive simulations. The simulation results demonstrate that, in various simulation scenarios, it outperforms the other four benchmark algorithms in terms of time-average task latency, time-average system execution efficiency, and time-average carbon emissions.

The remainder of this study is outlined as follows. Section 2 reviews related works. Section 3 describes the proposed green MEC-enabled MIoT network architecture, along with communication, computing and energy consumption, and carbon emissions models, followed by the problem formulation. Section 4 outlines the proposed algorithm, while

Section 5 discusses the results of the performance assessment. Finally, we conclude this paper in Section 6.

2. Related Works

Currently, existing maritime networks are dependent on narrowband radio transmissions, satellite links, and land-based cellular networks as the main methods of web access [26]. The bandwidth of the maritime radio system is restrictive. Despite the potential of satellite systems for furnishing vessels with global internet access, the prohibitive cost of the satellite terminals and service significantly hinders their popularity. Those living in coastal areas may reap the rewards of the growth of land-based cellular networks (e.g., 5G). In this study, the focus is on the issue of offloading computational tasks and organizing them for offshore cellular communication. There has been extensive research on the issue of computation offloading, though not in relation to maritime scenarios.

For terrestrial cellular networks, MEC is capable of significantly reducing the backhaul traffic, transmission cost, and data leakage risk of the network by offloading computational tasks to nearby servers. The authors in [27] have investigated the multi-user computational offloading problem in a single-server scenario using a Stackelberg game model under a software-defined network to achieve optimal task offloading. The model in [27] exhibits high computational complexity, and it does not apply to application scenarios related to real-time task offloading. For the real-time offloading application scenario, the problem was converted into a second-order cone programming problem and solved iteratively through an algorithm based on successive convex approximations, such that the solution method is simplified in [28]. However, [28] has not considered the dynamic mobility of edge nodes, leading to its weak applicability. To investigate the fundamental trade-off between latency and energy consumption in MEC systems, an iterative heuristic-based online offloading algorithm has been proposed in [29]. To minimize the system latency and determine the optimal offloading decision, the authors in [30] presented the computational offloading decision as a finite-time Markov decision process and employed a dynamic programming approach. However, the work in [30] does not consider the energy consumption constraints that can lead to high energy consumption by users. To meet the IoT's requirements for both delay and energy consumption, the authors in [31] proposed a framework based on decomposition and the continuous pseudo-convex method for the cooperative offloading problem between edge networks and central cloud computing in cellular networks that reduces the system cost by an iterative algorithm. MEC offloading scenarios actually have dynamic, stochastic, and time-varying characteristics, and the main purpose of the above task-offloading schemes for terrestrial networks is to obtain better offloading decisions, lacking consideration of the high real-time requirements of decision algorithms in actual scenarios.

As artificial intelligence technology is leaping forward, more and more researchers have progressively combined it with MEC technology, which can more effectively address the optimization problem of computing task offloading in dynamic, stochastic, and time-varying environments. The authors in [32] studied the task offloading and resource allocation problem in vehicular networks and proposed a Q-Learning-based task offloading and resource allocation algorithm, but only considered a single MEC system. To optimize the task processing delay while satisfying the energy constraint, the authors in [33] proposed a Q-Learning-based joint communication and computational resource allocation mechanism for multiple MEC systems in cellular networks and verified that the proposed method has better environmental adaptability through simulation experiments. In [34], a novel software-defined network (SDN) edge cloud-based Q-Learning optimization framework was adopted to formulate offloading decisions and resource allocation for dynamic offloading scenarios, which can quickly adapt to a communication environment with gradient updates and a small number of samples. Although the studies [32–34] have achieved good offloading results in some specific scenarios using Q-learning methods, when the state and action space of the optimization problem are too large and high-dimensional

continuous, the memory space for storing Q-values will grow exponentially, and a large time overhead will be incurred in searching for the optimal offloading decision. Thus, deep learning techniques have been employed to solve the problem of high dimensionality in the state space where conventional RL exists. Considering the task offloading decision problem in a multi-user and multi-server MEC scenario, the authors in [35] proposed an online offloading algorithm based on DRL to address the optimization problem of task offloading. In [36], a hybrid MEC platform including land-based vehicles and unmanned aerial vehicles (UAVs) was considered, and a hybrid online offloading algorithm based on deep learning was proposed to minimize the energy consumption of IoTDs. However, it is limited to one device's information input at a time, which does not apply to practical scenarios. The authors in [37] investigated the joint offloading problem between vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) and proposed a multi-intelligence DRL framework to achieve the goal of meeting the delay requirements of both V2I and V2V links. However, the convergence speed is slow.

In contrast to the above work on terrestrial networks, this study mainly focuses on the next-generation maritime information system, and a DRL-based offloading method is proposed for maritime MEC networks to support multiple maritime applications. Thus far, researchers have proposed several task offloading algorithms for maritime MEC, which conform to the requirements for low latency and high-reliability application services in maritime networks to a certain extent. To optimize the allocation mechanism of computational and communication resources under energy-limited and delay-sensitive conditions, the authors in [38] analyzed the trade-off between the delay and energy consumption of maritime communication and proposed a phased joint optimization algorithm. Considering the difference in network node density between the nearshore and farshore of the ocean, [39] proposed a task offloading model based on the nearshore and farshore scenarios, which was established and solved using a genetic algorithm and a particle swarm optimization algorithm, respectively. The authors in [40] used mixed-integer nonlinear programming to separate the optimization objectives, efficiently allocate the transmission power, and formulate the offloading decision by improving the conventional artificial fish swarm algorithm. However, the heuristic algorithm not only requires a large number of iterations during the optimization search but also the computational power of the algorithm decreases significantly in complex unloading environments and the solution quality cannot be guaranteed. Discretizing continuous variables with DQN or DDQN disrupts the continuity of the space, making it impossible to identify the optimal policy. In order to improve learning stability, DDPG leverages the experience replay buffer and target network strategies from DQN. The optimization of DDPG for resource allocation problems with continuous action space makes it a more efficient approach, and its training performance and dependability surpass those of the original actor-critic network. With joint consideration of energy consumption, task delay, and cache fetching cost while adhering to the limited storage and computational resources in MEC systems, the authors in [41] proposed a DDPG-based algorithm to optimize the long-term average system cost. Similarly, the authors in [42] proposed a temporal attentional deterministic policy gradient-based DDPG. Although the proposals in [41,42] have been shown to achieve faster and more reliable convergence compared to DQN, they consider energy savings more than carbon emission reductions.

In our initial work [25], we introduced a carbon-aware MEC framework for a hybrid renewable and grid-energy MEC system. We studied the problem of joint task offloading and resource allocation and proposed a DDPG-based joint optimization strategy considering stochastic tasks and renewable energy arrivals. The main objective was to minimize the total carbon emissions and task queue length. This current work includes several added contributions. First, we propose a new framework for green MEC-enabled maritime IoT networks and aim to optimize total system execution efficiency in an unpredictable environment. Second, we formulate the joint task offloading and resource allocation problem by maximizing system execution efficiency, which consists of the system execution cost (including system task latency and carbon emissions) and the size of completed tasks. Third,

we propose a DDPG-based joint optimization strategy, eventually obtaining an effective resolution through continuous action space learning in an unpredictable environment. Fourth, we made a detailed comparison of time-average system execution efficiency, time-average task latency, and time-average carbon emission with baseline strategies by investigating metrics in this study. Additionally, apart from the related work reviewed in our earlier work [25], this article further extensively summarizes and discusses the most pertinent studies on MEC offloading application scenarios, the algorithms employed, and the curse of dimensionality, with special attention to the optimization objectives, particularly in terms of carbon emissions.

Table 1 compares and summarizes the above literature through application scenarios, offloading algorithms, optimization objectives, and the curse of dimensionality. As depicted in Table 1, the conventional MEC offloading algorithms [27–31,38–40] have high computational complexity and are always less efficient than AI algorithms. Although the AI offloading algorithm based on RL [32–34] can achieve high offloading efficiency, there is a dimensional disaster that is only applicable to small-scale application scenarios. DRL-based offloading algorithms [35,36] are capable of overcoming the dimensionality catastrophe, although they are too focused on centralized offloading strategies and are less adaptable to new environments. As revealed by the comparison, the existing research on MEC for maritime networks has the following defects. To be specific, there has been limited research comprehensively considering task offloading and resource allocation in maritime edge collaborative architectures, and little work has taken into account reducing the carbon emissions of maritime networks by exploiting renewable energy. Furthermore, most existing works have slow convergence at high dimensionality and cannot conform to the requirements of low latency and high-reliability task offloading in maritime networks, and they are weakly adaptive to rapidly changing maritime environments.

Table 1. MEC offloading model comparison.

Ref.	Application Scenarios	Algorithms Applied	Dimensionality-Free Disaster	Optimization Objective		
				Carbon Emission	Latency	Task Execution Bits
[25]	Cellular	DDPG	√	√	√	×
[27]	Cellular	Stackelberg	×	×	×	×
[28]	Cellular	Successive approximation	×	×	×	√
[29]	Cellular	heuristic	×	×	√	×
[30]	Cellular	Dynamic planning	×	×	√	×
[31]	Cellular	Iterative	×	×	√	×
[32]	IoV	Q-learning	×	×	√	×
[33]	Cellular	Q-learning	×	×	√	√
[34]	Cellular	Q-learning	×	×	√	×
[35]	Cellular	DRL	√	×	√	×
[36]	Hybrid MEC	DNN	√	×	√	×
[37]	IoV	DRL	√	×	√	×
[38]	Maritime	Heuristic	×	×	√	×
[39]	Maritime	PSO	×	×	√	×
[40]	Maritime	AFSA	×	×	√	×
[41]	Cellular	DDPG	√	×	√	√
[42]	Cellular	DDPG	√	×	√	×
Our study	Maritime	DDPG	√	√	√	√

3. System Model

3.1. Network Architecture

In this study, we propose an offshore MEC architecture based on a renewable energy supply, which is shown in Figure 1. We consider an uplink transmission scenario for computation offloading in the MEC system with multiple cells and multiple MIoTDs, which consists of a multi-antenna macro-BS k deployed with an MEC server, a set of multiple

micro-BSs M = {1,2,. . ., M}, each equipped with a high-performance processor, energy access point (EAP), and energy storage unit (ESN), and a set of multiple MIoTDs (such as various sensors, smart buoys, and USVs), denoted as N = {1,2,. . ., N}. All MIoTDs are randomly distributed around micro-BSs, and the number of MIoTDs associated with micro-BS m composes a set N_m = {1,2,. . ., N_m}, $\forall N_m \in N$. Each micro-BS covers a certain area for control information delivery, which can sense the system channel state information (CSI) by requesting feedback information through the control link. All micro-BSs can connect to the macro-BS through the wireless links and generate control policies for computation offloading and resource allocations based on local CSI and computation task requirements. Each MIoTD can only communicate with an adjacent micro-BS and is unable to communicate directly with the macro-BS. However, all micro-BSs can establish communication with the macro-BSs through wireless channels. Any MIoTD contains an EH module and can draw energy from its linked micro-BS's EAP as long as its energy queue is not full. In each TS, the data generated by MIoTD can either be executed or dropped when its delay constraint cannot be satisfied. Depending on its own capability, MIoTD can be executed entirely locally or arbitrarily partially offloaded to its connected micro-BS or the macro-BS through the micro-BS according to the computation and energy resource. We assume that MIoTDs and micro-BSs can switch to conventional grid power when their ESNs' energy is insufficient, while the macro-BS always operates on grid energy. Table 2 lists the main notations (and their definitions) used in this paper.

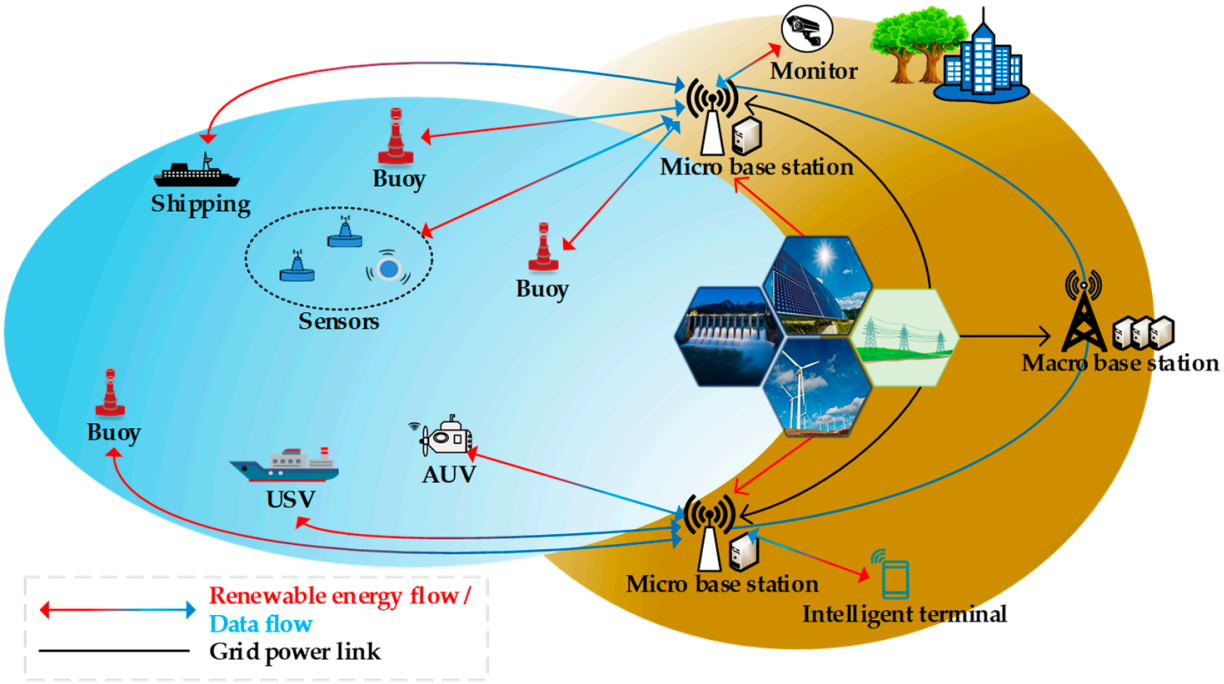

Figure 1. Illustration of a green MEC-enabled MIoT network architecture with hybrid energy.

Assuming that the macro-BS has sufficient computing resources, it can efficiently execute multiple parallel tasks without causing a queuing delay. Compared with the data offloaded to the macro-BS and micro-BSs, its execution result is significantly smaller. Thus, the transmission delay and energy consumption needed to send the output data back to the relevant MIoTDs can be ignored.

Table 2. List of main notations.

Notation	Definition
T	Index set of the time slots (TS)
τ	The length of each TS
k	The macro-BS
N	Set of MIoTDs
M	Set of micro-BSs or EAPs
$u_{m,n}(t)$	The computational task generated by MIoTD n associated with micro-BS m at TS t
$G_m(t)$	The energy harvesting capability of micro-BS m in TS t
$I_{m,n}(t)$	The CPU cycles needed to process per input data bit of $u_{m,n}(t)$
$C_{m,n}(t)$	The size of task $u_{m,n}(t)$
$T_{m,n}^{\max}(t)$	The maximum tolerable delay of $u_{m,n}(t)$
$\chi(t)$	The system execution efficiency in TS t
$p_{m,n}(t)$	The transmission power of MIoTD n for offloading to micro-BS m in TS t
$p_{k,m}(t)$	The transmission power of micro-BS m for offloading to macro-BS k in TS t
$p_{m,n}^{\max}$	The maximum uplink transmission power of MIoTD n associated with micro-BS m
$p_{k,m}^{\max}$	The maximum uplink transmission power of micro-BS m associated with macro-BS k
$x_{m,n}(t)$	Binary indicator of MIoTD n's task $u_{m,n}(t)$ in TS t
$\varphi_{m,n}(t)$	Drop mode indicator of MIoTD n's task $u_{m,n}(t)$ in TS t
$DE_{m,n}(t)$, $TE_{m,n}(t)$, $EE_{m,n}(t)$	The local computing bits/computation latency/energy consumption of MIoTD n in TS t
$f_{m,n}(t)$	The CPU cycles per second for executing tasks locally on MIoTD n associated with micro-BS m in TS t
$f_{m,n}^{\max}$	The maximum computing capacity of MIoTD n associated with micro-BS m
$f_m(t)$, $f_k(t)$	The CPU cycles per second on micro-BS m/macro-BS k for the tasks from MIoTDs
$f_m^{\max}$	The maximum computing capacity of micro-BS m
$f_k^{\max}$	The maximum computing capacity of macro-BS k
$DT_{m,n}(t)$, $TT_{m,n}(t)$, $ET_{m,n}(t)$	The sum of each MIoTD n's computing task bits/transmission latency/energy consumption transmitted to micro-BS m in TS t
$\xi_m(t)$	The offloading data ratio of micro-BS m to macro-BS k in TS t
$DE_m(t)$, $TE_m(t)$, $EE_m(t)$	The sum of all MIoTDs' computing bits/computing latency/energy consumption executed in micro-BS m in TS t
$DE_k(t)$, $TE_k(t)$, $EE_k(t)$	The number of computing bits/computing latency/energy consumption processed in the macro-BS k in TS t
$TT_{k,m}(t)$, $ET_{k,m}(t)$	The latency/energy consumption for uploading data from micro-BS m to macro-BS k in TS t
$D(t)$, $T(t)$, $CER(t)$	The total number of computing bits/overall system latency/carbon emissions of system in TS t
w	The tradeoff weight between $T(t)$ and $CER(t)$
$COST(t)$	The weight sum system execution cost of $T(t)$ and $CER(t)$ in TS t
$E_m(t)$	The energy usage of micro-BS m in TS t
$g_m(t)$	The ratio of green energy consumed by micro-BS m to its total consumed energy in TS t
$B_m(t)$	The dynamics of the renewable energy level of each EH module in micro-BS m in TS t

3.2. Communication Model

The consecutive duration of the process is divided into separate time slots (TSs), wherein each period τ consistently remains at 2 ms. The OFDM access approach has been adopted to reduce inter-MIoTD interference within the same micro-BS. The available bandwidth of the micro-BS m, represented as W_m, is divided into N_m subchannels of equal width, where N_m is the number of MIoTDs connected to the micro-BS m at TS t. Then, the

bandwidth of each MIoTD subchannel can be calculated as $BW_m = W_m/N_m$. Therefore, the achievable data rate between MIoTD n and its associated micro-BS m is denoted as

$$r_{m,n}(t) = BW_m \log_2\left(1 + \frac{p_{m,n}(t)h_{m,n}(t)}{BW_m N_{m,n}}\right), \tag{1}$$

where $p_{m,n}(t)$, $h_{m,n}(t)$, and $N_{m,n}$ denote the transmit power of MIoTD n, the channel gain between MIoTD n and micro-BS m, and the power spectrum density of the additive white Gaussian noise (AWGN), respectively. Let the maximum allowable transmission power of MIoTD n be denoted as $p_{m,n}^{\max}$, and $p_{m,n}(t) \leq p_{m,n}^{\max}$. Similarly, the available bandwidth W_k of macro-BSs is divided into M equal-width subchannels for micro-BSs. The bandwidth of each subchannel assigned to a micro-BS can be formulated as $BW_k = W_k/M$. Hence, the achievable upload data rate between micro-BS m and macro-BS k can be represented as

$$r_{k,m}(t) = BW_k \log_2\left(1 + \frac{p_{k,m}(t)h_{k,m}(t)}{BW_k N_{k,m}}\right), \tag{2}$$

where $h_{k,m}(t)$ and $N_{k,m}$ express the channel gain between micro-BS m and macro-BS k, as well as the power spectrum density of the AWGN, and $p_{k,m}(t)$ refers to the transmitting power of micro-BS m for offloading to the macro-BS k. Let the maximum allowable transmission power of micro-BS m be $p_{k,m}^{\max}$ and $p_{k,m}(t) \leq p_{k,m}^{\max}$.

3.3. Computing and Energy Consumption Model

In this portion, we illustrate the computation offloading and energy consumption models that we have studied. It is assumed that each MIoTD possesses restricted processing capacities, which can be used to handle the requested computation tasks locally. We consider a stochastic task arrival model in which the computation task of MIoTD n is defined as $u_{m,n}(t) = \{I_{m,n}(t), C_{m,n}(t), T_{m,n}^{\max}(t)\}$, with $I_{m,n}(t)$ as the number of CPU cycles required for computing 1-bit data (cycles per bit) and $C_{m,n}(t)$ as the data size (kbits) of the task. $T_{m,n}^{\max}(t)$ is the maximum delay tolerance (seconds) of $u_{m,n}(t)$. For task $u_{m,n}(t)$, the computation capability required is $I_{m,n}(t)\, C_{m,n}(t)/T_{m,n}^{\max}(t)$ (cycles per second). In each TS, each computation task can either be executed or dropped when its delay constraint cannot be satisfied. $\varphi_{m,n}(t)$ denotes the drop mode indicator. $\varphi_{m,n}(t) = 0$ indicates the computation task is dropped, while $\varphi_{m,n}(t) = 1$ means that the task is executed in the TS. Depending on its own capability, MIoTD n can opt for either local computing of the task or offloading it entirely to its connected micro-BS or partially to the macro-BS through the micro-BS. For each TS t, let $x_{m,n}(t)$ be a binary indicator; that is, 1 if MIoTD n has a task $u_{m,n}(t)$ to be offloaded and 0 otherwise (i.e., locally executed).

3.3.1. Local Computing

When task $u_{m,n}(t)$ is chosen for local processing, the processing performance is determined by the quality of MIoTD n's computing capability, which can be characterized by the CPU-cycle frequency $f_{m,n}(t)$. The computation latency of MIoTD n can therefore be presented as follows:

$$TE_{m,n}(t) = \frac{DE_{m,n}(t) \times I_{m,n}(t)}{f_{m,n}(t)}, \tag{3}$$

where $DE_{m,n}(t)$ is the local computing bits of the MIoTD allocated in terms of local execution in TS t, which can be expressed as

$$DE_{m,n}(t) = \varphi_{m,n}(t) \times (1 - x_{m,n}(t)) \times C_{m,n}(t). \tag{4}$$

As defined in [43], the energy consumption to execute the tasks locally can be denoted as

$$EE_{m,n}(t) = k_a \times f_{m,n}^2(t) \times DE_{m,n}(t), \tag{5}$$

where k_a is a nonnegative coefficient depending on the chip architecture, and k_b and k_c mentioned later are similar to k_a.

When MIoTDs lack the computing capacities to complete the tasks locally, the tasks are offloaded to the micro-BS for processing. Specifically, the MIoTD n's computation task is initially transmitted to the closest micro-BS m via a wireless connection for processing. If the micro-BS m lacks adequate computation resources or energy to complete the task, it can offload the computation task to macro-BS k, which is equipped with more computation resources, and then retrieve the result once the task is complete.

According to the above, we can obtain the sum of the computing task bits in MIoTD n transmitted to micro-BS m at TS t, which is

$$DT_{m,n}(t) = \min(r_{m,n}(t) \times \tau, \varphi_{m,n}(t) \times x_{m,n}(t) \times C_{m,n}(t)). \tag{6}$$

when the offloaded data size is more than the maximum number of task bits that can be transmitted in TS t between micro-BS m and MIoTD n, the MIoTD does not offload the tasks, i.e., $\varphi_{m,n}(t) = 0$. Correspondingly, the transmission latency and energy consumption for $DT_{m,n}(t)$ are, respectively,

$$TT_{m,n}(t) = \frac{DT_{m,n}(t)}{r_{m,n}(t)}, \tag{7}$$

$$ET_{m,n}(t) = p_{m,n}(t) \times TT_{m,n}(t). \tag{8}$$

3.3.2. Edge Computing

As aforementioned, suppose each computing task can only be offloaded to one micro-BS, is fine-grained, and can be arbitrarily split into two parts in the micro-BS: one part is executed on its high-performance processor, while the other is offloaded and processed on the MEC server in the macro-BS. Let $\xi_m(t)$ denote the offloading data ratio of micro-BS m to macro-BS k. The micro-BS m executes $\xi_m(t) \times DT_{m,n}(t)$, and the amount of computing bits executed by the macro-BS k is $(1 - \xi_m(t)) \times DT_{m,n}(t)$. The sum of all MIoTDs' computing bits executed in micro-BS m at TS is

$$DE_m(t) = \xi_m(t) \times \sum_{n=1}^{N_m} DT_{m,n}(t). \tag{9}$$

The computing tasks transmitted by MIoTDs can be given recourse to the CPU cycles provided by the high-performance processor of micro-BS m. The needed CPU cycles for computing tasks from the MIoTDs connected to the m-th high-performance processor should be no larger than the available CPU cycles of the m-th high-performance processor. Thus, we have

$$\xi_m(t) \times \sum_{n=1}^{N_m} [DT_{m,n}(t) \times I_{m,n}(t)] \leq f_m^{\max} \tau, \tag{10}$$

where $f_m^{\max}$ refers to the m-th high-performance processor's maximization available computation capability and τ refers to the length of each TS. The related computing latency and energy consumption of each micro-BS are

$$TE_m(t) = \frac{\xi_m(t) \times \sum_{n=1}^{N_m} [DT_{m,n}(t) \times I_{m,n}(t)]}{f_m(t)}, \tag{11}$$

$$EE_m(t) = k_b \times f_m^2(t) \times DE_m(t), \tag{12}$$

where $f_m(t)$ denotes the computation resources provided by the micro-BS m for the tasks from MIoTDs.

When computing tasks are offloaded partially to the macro-BS, the number of computing bits processed in the macro-BS in TS t is

$$DE_k(t) = \sum_{m=1}^{M} \sum_{n=1}^{N_m} [(1 - \xi_m(t)) \times DT_{m,n}(t)]. \tag{13}$$

Similarly, the computation requirements of the MIoTDs cannot surpass the MEC server's available computation capability. Thus, we have

$$\sum_{m=1}^{M} \sum_{n=1}^{N_m} [(1 - \xi_m(t)) \times DT_{m,n}(t) \times I_{m,n}(t)] \leq f_k^{\max} \tau. \tag{14}$$

where $f_k^{\max}$ refers to the MEC server's maximization available computation capability and τ refers to the length of each TS. Let $f_k(t)$ denote the computation resources provided by the macro-BS k for the tasks from MIoTDs. The computing latency and energy consumption of $DE_k(t)$ are

$$TE_k(t) = \frac{\sum_{m=1}^{M} \sum_{n=1}^{N_m} [(1 - \xi_m(t)) \times DT_{m,n}(t) \times I_{m,n}(t)]}{f_k(t)}, \tag{15}$$

$$EE_k(t) = k_c \times {f_k}^2(t) \times DE_k(t). \tag{16}$$

The latency and energy consumption for uploading data from the micro-BS m to the macro-BS k are

$$TT_{k,m}(t) = \frac{DE_k(t)}{r_{k,m}(t)}, \tag{17}$$

$$ET_{k,m}(t) = p_{k,m}(t) \times TT_{k,m}(t). \tag{18}$$

Based on the task execution at various offloading modes evidenced above, the total number of computing bits that have been handled by the system in TS t is

$$D(t) = \sum_{m=1}^{M} \sum_{n=1}^{N_m} DE_{m,n}(t) + \sum_{m=1}^{M} DE_m(t) + DE_k(t). \tag{19}$$

The overall system latency includes the latency of the local execution of tasks and the latency of remote transmission and execution in the system at TS t, which can be represented as follows:

$$T(t) = \sum_{m=1}^{M} \sum_{n=1}^{N_m} TE_{m,n}(t) + \sum_{m=1}^{M} \sum_{n=1}^{N_m} TT_{m,n}(t) + \sum_{m=1}^{M} TE_m(t) + \sum_{m=1}^{M} TT_{k,m}(t) + TE_k(t). \tag{20}$$

Correspondingly, the time-average task latency of the system can be denoted as follows:

$$TATL = \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T} T(t). \tag{21}$$

According to [16], the energy consumption of micro- or macro-BS can be attributed to two components: static power offset and energy consumption for task execution or data transmission. Since their static power consumption is the same for different offloading schemes, it is negligible in the following model. The energy usage of micro-BS m encompasses the energy consumption related to executing tasks locally and transmitting data to the macro-BS, as well as charging power to its associated MIoTDs. Let η denote the inverse of the power amplifier efficiency factor for the process of micro-BS to charge an MIoTD. Assuming equal power amplifier efficiency factor (η) for all MIoTDs, the amount of energy used by a micro-BS to charge an MIoTD is η times the quantity of energy consumed by the

MIoTD to locally complete the task and transmit its offloaded task to the micro-BS. Hence, the energy consumption of micro-BS m can be expressed as

$$E_m(t) = EE_m(t) + \eta \sum_{n=1}^{N_m} [EE_{m,n}(t) + ET_{m,n}(t)] + ET_{k,m}(t). \tag{22}$$

3.4. Carbon Emission Model

It is suggested that the energy-harvesting capability of micro-BS m at TS t is $G_m(t)$. For realistic considerations, the collected green energy is not the sole source of energy for each micro-BS and all MIoTDs in the network, which can also use traditional power when the harvested energy cannot fulfill their energy requirements. Let $g_m(t)$ represent the ratio of green energy used by micro-BS m to its total consumed energy $E_m(t)$. Therefore, the green energy used by each micro-BS can be expressed as $E_m(t)g_m(t)$. Considering the energy level of the EH module in micro-BS m at TS t, we have

$$E_m(t)g_m(t) \leq B_m(t). \tag{23}$$

Therefore, the dynamics of the renewable energy level $B_m(t)$ of each EH module in micro-BS m is symbolized as

$$B_m(t+1) = \max[B_m(t) - E_m(t) + G_m(t), 0]. \tag{24}$$

when $B_m(t) > 0$, micro-BS m operates on renewable energy sources. When $B_m(t) \leq 0$, micro-BS m and its covered MIoTDs switch to a conventional energy supply due to insufficient renewable energy. It is clear that the second scenario would result in higher carbon emissions of the system.

Based on the above, for each TS t, the network's energy consumption may consist of grid power and renewable energy. Hence, the system's grid power usage can be represented as

$$E(t) = EE_k(t) + \sum_{m=1}^{M} E_m(t)(1 - g_m(t)). \tag{25}$$

Hence, the system's total carbon emissions and time-average carbon emissions can be respectively expressed as

$$CER(t) = E(t) \times \varepsilon, \tag{26}$$

$$TACE = \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T} CER(t), \tag{27}$$

where ε is the carbon emission factor in kg/kWh, i.e., the amount of CO_2 released when using 1 kWh of conventional energy, typically set at 0.998 kg/kWh [25].

We defined the total system execution cost as

$$COST(t) = w \times T(t) + (1 - w) \times CER(t), \tag{28}$$

In actuality, $COST(t)$ is a compromised tradeoff between the task latency and carbon emissions of the system at TS t. w is the weight coefficient. If w is sufficiently large, it demonstrates that MIoTD n is delay-sensitive with much lower latency. Otherwise, the importance of reducing traditional energy consumption and carbon emissions can be significantly highlighted.

3.5. Problem Formulation

In this subsection, the joint optimization problem of computation offloading and resource allocation is formulated under the uplink MIoT network scenario. We defined

the system execution efficiency as the ratio of system execution cost to the total size of completed tasks, which can be expressed as

$$\chi(t) = \frac{D(t)}{COST(t)}, \tag{29}$$

Further, we defined the time-average system execution efficiency as

$$TASEE = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} \chi(t), \tag{30}$$

Our aim is to reduce the system's carbon emissions and task delay and improve the total size of completed tasks, that is, maximize system execution efficiency while fulfilling the constraints of the task requirements, which can be formulated as follows:

$$\max \frac{1}{T} \sum_{t=0}^{T} \chi(t), \tag{31a}$$

s.t.

$$0 \le g_m(t) \le 1, \forall m \in M \tag{31b}$$

$$\varphi_{m,n}(t) \in \{0,1\}, x_{m,n}(t) \in \{0,1\}, \forall m \in M, n \in N \tag{31c}$$

$$0 \le p_{m,n}(t) \le p_{m,n}^{\max}, 0 \le p_{k,m}(t) \le p_{k,m}^{\max}, \forall m \in M, n \in N \tag{31d}$$

$$0 \le f_{m,n}(t) \le f_{m,n}^{\max}, \forall m \in M, n \in N \tag{31e}$$

$$0 \le f_m(t) \le f_m^{\max}, 0 \le f_k(t) \le f_k^{\max}, \forall m \in M \tag{31f}$$

$$\xi_m(t) \times \sum_{n=1}^{N_m} [DT_{m,n}(t) \times I_{m,n}(t)] \le f_m^{\max}\tau, \ \forall m \in M, \ n \in N \tag{31g}$$

$$\sum_{m=1}^{M} \sum_{n=1}^{N_m} [(1 - \xi_m(t)) \times DT_{m,n}(t) \times I_{m,n}(t)] \le f_k^{\max}\tau, \forall m \in M, n \in N \tag{31h}$$

$$E_m(t) g_m(t) \le B_m(t), \forall m \in M \tag{31i}$$

$$0 \le \xi_m(t) \le 1, \forall m \in M \tag{31j}$$

$$T_{m,n}(t) \le T_{m,n}^{\max}(t), \forall m \in M, n \in N \tag{31k}$$

where (31b) indicates the restriction on the ratio of green energy consumed. (31c) is the zero-one constraint for the drop and computation mode indicators of task $u_{m,n}(t)$ at TS t. (31d) represents the transmission power constraint in terms of MIoTDs and micro-BSs. (31e) and (31f) represent the CPU-cycle frequency constraint of any MIoTD, the high-performance processor, and the MEC server. (31g) and (31h) are the scheduling restrictions of each micro-BS and the macro-BS. (31i) guarantees that the green energy used by micro-BS m does not exceed its renewable energy level in TS t. (31j) denotes the constraint of the offloading data ratio from micro-BS m to macro-BS k. (31k) ensures the latency requirements of each task. $T_{m,n}(t)$ refers to the total latency of the task $u_{m,n}(t)$ at TS t. Task $u_{m,n}(t)$ may be executed either fully locally or offloaded. For the latter case, the task may be completed entirely at the micro-BS, or partially at the micro-BS, and the other part is executed at the macro-BS.

Therefore, $T_{m,n}(t)$ includes the local execution delay or the transmission and execution delay during offloading and can be represented as follows:

$$T_{m,n}(t) = TE_{m,n}(t) + DT_{m,n}(t) \times \left[\frac{I_{m,n}(t)\xi_m(t)}{f_m(t)} + \frac{1}{r_{m,n}(t)} + \frac{I_{m,n}(t)(1-\xi_m(t))}{f_k(t)} + \frac{1-\xi_m(t)}{r_{k,m}(t)} \right]. \tag{32}$$

Considering the uncertainty of computational tasks and renewable energy in problem (31), next, we employ DRL techniques to determine the optimal offloading policy.

4. DDPG-Based Algorithm Design for Task Offloading

4.1. Problem Solution by DDPG

In the proposed MEC model, each MIoTD can choose either to execute its task locally or offload the task to its connected micro-BS or the macro-BS via the micro-BS. The arrival of computing tasks and renewable energy sources is unpredictable and wireless channel is time-varying. Therefore, we utilize the DRL technique to learn the optimization computing offloading policy considering stochastic tasks, renewable energy, and wireless channel conditions. We employ the DDPG algorithm to continuously learn the optimal scheme for local execution and task offloading, utilizing a continuous action space. This approach is an improvement upon DQN [44]. DQN [45] serves as the baseline algorithm of DRL, and it has seen widespread utilization across many optimization areas. Despite this, the action spaces of DQN are discrete, while the proposed MEC model's continuous action space is required to be discretized when DQN is chosen. If the action space is excessively vast, the problem of the high-dimensional will emerge. By comparison, DDPG is an extension of the actor-critic algorithm, which can effectively manage a continuous action space [46]. The actor network modifies the neural network's parameters, utilizing a deterministic policy gradient to identify the optimal action in the current state. The critic network accesses the actor network's policy by measuring the time difference error. The critic network adjusts the neural network's parameters by utilizing the Q-function. Rather than using a traditional stochastic policy gradient, DDPG selects actions based on the action distribution. By employing DDPG, a smaller number of data are sampled, which can enhance the algorithm's efficiency. DDPG leverages the architecture of DQN in its critic network and employs an off-policy approach to achieving a balance between exploration and exploitation.

Figure 2 illustrates that DDPG applies two separate DNNs to model the actor network $\mu(s \mid \theta^{\mu})$, i.e., the policy function, and the critic network $Q(s, a \mid \theta^{Q})$, i.e., the Q-value function, respectively. Moreover, the actor and critic networks have a corresponding target network with an identical structure: A critic target network Q' with parameters $\theta^{Q'}$ and an actor target network μ' with parameters $\theta^{\mu'}$. Analogous to DQN, the critic network $Q(s, a \mid \theta^{Q})$ is updated as follows:

$$L\left(\theta^{Q}\right) = E_{\mu'}\left[\left(y_i - Q\left(s_i, a_i \mid \theta^{Q}\right)\right)^2\right], \tag{33}$$

where

$$y_i = r_i + \gamma Q\left(s_{i+1}, \mu(s_{i+1}) \mid \theta^{Q}\right). \tag{34}$$

Silver et al. [47] have demonstrated that the policy gradient can be revised using the chain rule.

$$\nabla_{\theta^{\mu}} J \approx E_{\mu'}\left[\nabla_{\theta^{\mu}} Q\left(s, a \mid \theta^{Q}\right) \mid_{s=s_i, a=\mu(s_i \mid \theta^{\mu})}\right] = E_{\mu'}\left[\nabla_a Q\left(s, a \mid \theta^{Q}\right) \mid_{s=s_i, a=\mu(s_i \mid \theta^{\mu})} \nabla_{\theta^{\mu}} \mu(s \mid \theta^{\mu}) \mid_{s=s_i}\right]. \tag{35}$$

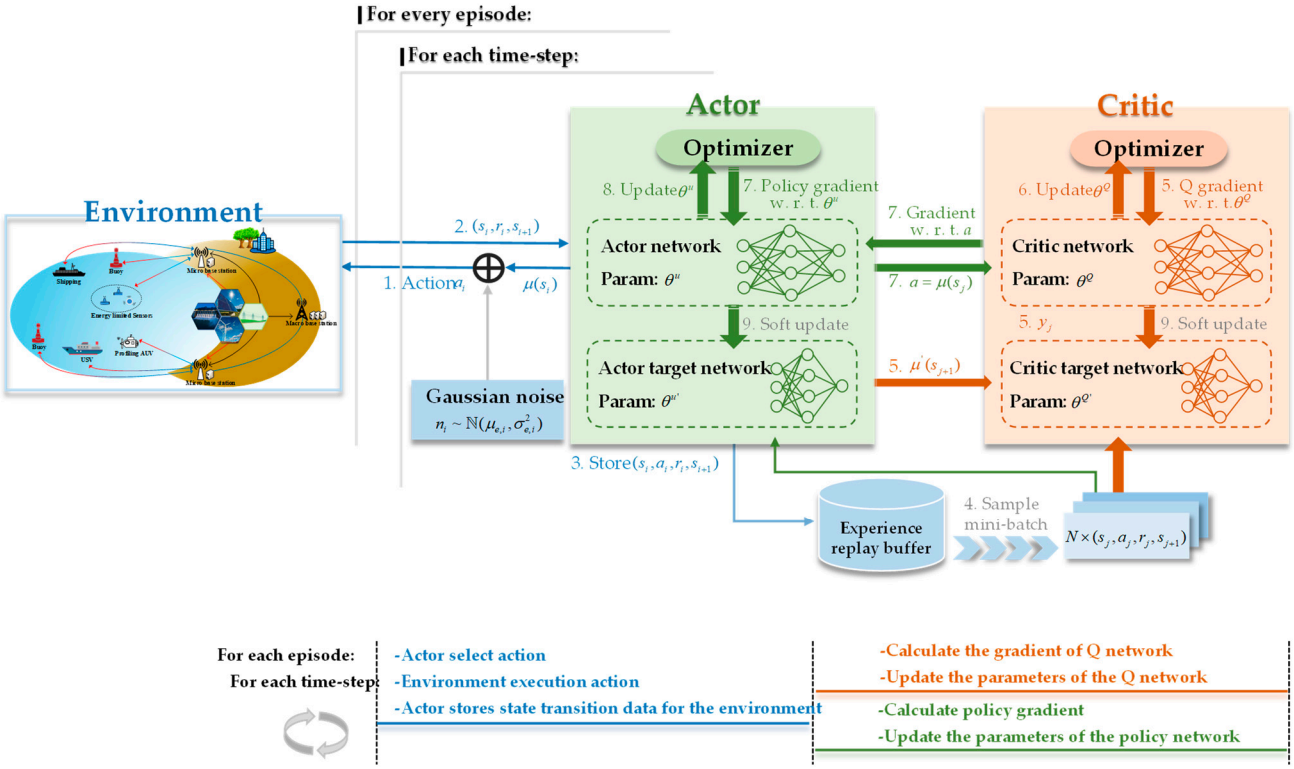

Figure 2. Structure of the DDPG-based joint computation offloading and resource allocation algorithm.

The DDPG algorithm's entire training process is concluded in the following. The correlation between exploration and exploitation should be assessed after the actor network μ produces $\mu(s_i)$ from the prior training stage to ensure a thorough exploration of the state space. We can conduct the investigation of DDPG independently of the training procedure since DDPG refers to an off-policy algorithm. Consequently, the action space is created by incorporating behavior noise n_i for obtaining action $a_i = \mu(s_i) + n_i$, for which n_i follows a Gaussian distribution n_i~$\mathbf{N}(\mu_e, \sigma^2_{e,i})$, with $\sigma_{e,I}$ as the standard deviation and μ_e as the mean. Upon executing a_t in the environment, the agent will be able to witness s_{i+1} as the subsequent state and be rewarded with r_t. Next, the experience replay buffer stores the transition (s_i, a_i, r_i, s_{i+1}). Subsequently, the algorithm randomly selects N transition (s_j, a_j, r_j, s_{j+1}) from the buffer, such that a mini-batch can be established. Next, the established mini-batch is introduced into the actor network and the critic network. The actor target μ' network produces the action $\mu'(s_{j+1})$. Afterward, $\mu'(s_{j+1})$ is sent to the critic target network Q' via the mini-batch. Using mini-batch and $\mu'(s_{j+1})$, the critic network can determine the target value y_j in accordance with (35).

The Q critic network should be modified using an optimizer (e.g., the Adam Optimizer) for a reduction in the loss function. Subsequently, the actor network μ supplies the minibatch action $a = \mu(s_j)$ to the critic network to acquire the action's gradient $\nabla_a Q(s, a \mid \theta^Q) \mid_{s=sj,\ a=\mu(sj)}$. The value of $\nabla_{\theta\mu}\mu(s \mid \theta^{\mu}) \mid_{s=sj}$ can be determined using its optimizer. Using the above-described two gradients, the actor network can be updated with the following approximation:

$$\nabla_{\theta^{\mu}} J \approx \frac{1}{N}\sum_{j}\left[\nabla_a Q\left(s, a \mid \theta^{Q}\right) \mid_{s=s_j, a=\mu(s_j \mid \theta^{\mu})} \nabla_{\theta^{\mu}} \mu(s \mid \theta^{\mu}) \mid_{s=s_j}\right]. \tag{36}$$

Lastly, the DDPG agent applies a minor fixed value τ_0 to gradually update the actor target network and the critic target network, separately

$$\theta^{Q\prime} \leftarrow \theta^{Q} + (1 - \tau_0)\theta^{Q\prime}, \tag{37}$$

$$\theta^{\mu\prime} \leftarrow \theta^{\mu} + (1 - \tau_0)\theta^{\mu\prime}. \tag{38}$$

4.2. DDPG-Based Algorithm

In this section, we first thoroughly define the core components of the DRL agent, including specific states, actions, and rewards. Then we propose a DDPG-based joint computation offloading and resource allocation algorithm to find the most cost-effective policy adaptively. The agent retrieves the current micro-BS state s_t by regularly communicating with the associated micro-BS. This agent then generates an action at based on s_t via a strategy, for example, a deterministic policy, as guidance. Once action a_t is taken in micro-BS m, the reward $r_t(s_t, a_t)$ is provided instantly. By finding the best action policy, the agent aims to maximize the long-term reward R_t in a decision episode. The state space, action space, and reward function are outlined as follows.

4.2.1. State Space

At TS t, the status of micro-BS m contains five parameters, including attributes related to the task and environment variables. The task attributes include the requested task profiles of task $u_{m,n}(t)$ (including the size of the computation task $C_{m,n}(t)$, the maximum tolerable delay $T_{m,n}^{\max}(t)$, and the number of CPU cycles needed to process per input data bit of $I_{m,n}(t)$). The size of the energy packet $G_m(t)$ and the energy level of micro-BS m $B_m(t)$ at t-th TS and the system state s_t at TS t can be described as

$$s_t = \{C_{m,n}(t), I_{m,n}(t), T_{m,n}^{\max}(t), G_m(t), B_m(t)\}. \tag{39}$$

4.2.2. Action Space

Given the current state and observed environment of the system, the action a_t can be defined as

$$a_t = \{x_{m,n}(t), \xi_m(t), g_m(t), f_{m,n}(t), f_m(t), f_k(t), p_{m,n}(t), p_{k,m}(t)\}. \tag{40}$$

Clearly, a_t is a direct solution to the MIoTD's task of dynamically offloading computations and allocating resources. The action space for MIoTD's task $u_{m,n}(t)$ encompasses the possible actions for the variables: $x_{m,n}(t) \in \{0,1\}$ represents the available offloading modes to execute the computation task, $\xi_m \in [0,1]$, $g_m(t) \in [0,1]$, the uplink transmit power of each MIoTD and micro-BS, represented as $p_{m,n}(t) \leq p_{m,n}^{\max}$ and $p_{k,m}(t) \leq p_{k,m}^{\max}$, the computation resource allocated to each MIoTD, micro-BS and macro-BS, denoted as $f_{m,n}(t) \leq f_{m,n}^{\max}$, $f_m(t) \leq f_m^{\max}$, and $f_k(t) \leq f_k^{\max}$. Variables in a continuous action space can be optimized precisely and jointly.

4.2.3. Reward Function

The agent interacts with the environment using s_t and decides on a_t to receive an immediate reward $r_t(s_t, a_t)$, which is then added to the cumulative reward from the previous state $s_{t\text{-}1}$. Generally, the reward function is usually associated with the objective function. We define the system execution efficiency from Equation (29) as the immediate reward upon completing a series of actions for requested tasks, and it can be depicted as follows:

$$r_t(s_t, a_t) = \chi(t). \tag{41}$$

Consequently, the learning process aims to minimize the overall system execution cost while improving the size of completed tasks in order to maximize the expected cumulative rewards. We define the long-time reward value R_t of micro-BS m as

$$R_t = \max\left[\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} (\gamma_t \cdot r_t(s_t, a_t))\right], \tag{42}$$

where γ_t is the discount factor to determine how much importance should be given to the immediate reward r_t (s_t, a_t) that is received over time, with $0 \leq \gamma_t \leq 1$. A desirable reward achieves a higher return of R_t, and the goal of our agent is to maximize the value of R_t within a period of time.

4.2.4. Algorithm Description and Complexity Analysis

Herein, because discrete and continuous variables are intermingled and continuous variables have infinite values, the DQN scheme cannot be utilized directly. The size of the action space increases exponentially as the discrete level expands, leading to a reduction in performance since discretizing a continuous variable into a finite level will result in quantization error [48]. To address the problem of continuous variables, we developed the DDPG-based algorithm. As previously mentioned, each micro-BS stores computation-intensive tasks offloaded by MIoTDs and determines the offloading ratio for each task to the macro-BS. We illustrate the training process for the agent on micro-BS m, as depicted in Algorithm 1. The target networks for the critic and actor networks are replicas of their corresponding online networks (Step 2). Steps 10–17 outline the details of the DCTORA method. In steps 18 and 19, the Adam optimizer and soft update are employed to adjust the weight vectors of the critic and actor networks, both for the online and target networks. The training stops once the pre-set number of episodes, i.e., $E^{\max}$, has been completed.

Algorithm 1: Joint optimization method based on the proposed DCTORA

1: Randomly initialize the weights of actor network θ^{μ} and critic network θ^{Q}, respectively.
2: Initialize the target network with weights $\theta^{\mu} \leftarrow \theta^{\mu'}$ and $\theta^{Q} \leftarrow \theta^{Q'}$, respectively.
3: Empty the experience replay buffer $\boldsymbol{D}$.
4: **for** episode = 1 **to** $E^{\max}$ **do**
5: **for** t = 1 **to** T **do**
6: Reset simulation parameters of the system and obtain initial observation state s_1.
7: Normalize state s_t to $s_{t'}$.
8: Get the action with actor network θ^{μ} and perform action a_t according to (40).
9: Obtain the reward r_t according to (41) and observe the next state s_{t+1}.
10: **if** the replay buffer is not full **then**
11: Store transition $(s_{t'}, a_t, r_t, s_{t'+1})$ in replay buffer $\boldsymbol{D}$.
12: **else**
13: Randomly replace a transition in replay buffer $\boldsymbol{D}$ with $(s_{t'}, a_t, r_t, s_{t'+1})$.
14: Randomly sample N transition tuples from experience replay memory M as mini-batch data for training the main network of the actor and the critic.
15: Calculate the gradient $\nabla_{\theta^{Q}} L$ of the critic's main network according to (33) and (34).
16: Update the parameters θ^{Q} of the critic's main network using the Adam optimizer.
17: Calculate the policy gradient $\nabla_{\theta^{\mu}} L$ of the actor's main network according to (35).
18: Update the parameter θ^{μ} of the actor's main network using the Adam optimizer.
19: Soft update the parameters $\theta^{Q'}$, and $\theta^{\mu'}$ of the actor's target network and the critic's target network according to (37) and (38), where $\tau_0 = 0.001$.
20: **end if**
21: **end for**
22: **end for**

We analyze the proposed algorithm's time complexity in terms of floating-point operations per second (FPOPS). The time complexity of Algorithm 1 is largely dependent on the number of MIoTDs and the neural network structure employed for implementing the DDPG networks of each micro-BS agent. We assume that each actor network and each critic network comprise K and H fully connected layers, respectively. Moreover, $v_{actor,k}$ and $v_{critic,h}$ are the input sizes of the k-th layer actor network and the h-th layer critic network. Each layer in the actor network has a vector $v_{actor,k}$ and a matrix $v_{actor,k} \times v_{actor,k+1}$ to perform the dot product. Similarly, each layer in the critic network has a vector $v_{critic,h}$ and a matrix $v_{critic,h} \times v_{critic,h+1}$ to perform the dot product. According to the FPOPS, the computation is

$(2v_{actor,k} - 1) \times v_{actor,k+1}$, and every column in the matrix needs to be multiplied $v_{actor,k}$ times and added $v_{actor,k} - 1$ times. Therefore, the time complexity of Algorithm 1 is formulated as

$$\begin{aligned} &2\sum_{k=0}^{K-1}((2v_{actor,\,k}-1)\cdot v_{actor,\,k+1}+\varpi v_{actor,\,k+1})+2\sum_{h=1}^{H-1}((2v_{critic,\,h}-1)\cdot v_{critic,\,h+1}+\varpi v_{critic,\,h+1}) \\ &=O\left(\sum_{k=0}^{K-1}v_{actor,\,k}\cdot v_{actor,\,k+1}+\sum_{h=0}^{H-1}v_{critic,\,h}\cdot v_{critic,\,h+1}\right) \end{aligned} \tag{43}$$

where ϖ refers to the coefficient that is based on the type of activation layer used in the corresponding network.

5. Simulation Results and Analysis

In this section, we evaluate the performance of our proposal through extensive simulations using Matlab 2020a to illustrate how well it performs in the MEC-enabled MIoT system for computation offloading. Initially, the simulation parameters are outlined. The performance of the proposed algorithm (DCTORA) is then compared to four benchmark algorithms described below in terms of the time-average cumulative reward, time-average system execution efficiency, time-average task latency, and time-average carbon emissions across different scenarios. For convenience, we will abbreviate "time-average cumulative reward", "time-average system execution efficiency", "time-average task latency" and "time-average carbon emission" as "cumulative reward", "execution efficiency", "task latency" and "carbon emission", respectively, in this section:

- Full local execution (FL): All MIoTDs independently perform their tasks using their local computing resources.
- Full offloading (FO): Each MIoTD offloads its tasks to its connected micro-BS with the allocated transmit power or transmits the tasks to the macro-BS through the micro-BS; that is, all computation tasks are executed by utilizing the computation resources of the micro-BS's high-performance processors or the MEC server of the macro-BS.
- Greedy policy (GP): The system utilizes its maximum power to complete computing tasks either locally or remotely, completing tasks as often as possible.
- DQN-based offloading Scheme (DOS): Each agent adopts the DQN approach to train its model for joint optimization of task offloading and resource allocation, in which computing resource and power allocation are separated into 10 levels with values ranging from 0 to their maximum values.

5.1. Simulation Settings

In our simulations, we consider a system with multiple cells each with a micro-BS, and each micro-BS coverage radius is 600 m, where the macro-BS is located at the center of the micro-BS and multi-MIoTDs are randomly distributed between [50, 200] m from each micro-BS and [200, 1000] m from the macro-BS. The length of each TS is set to 2 ms, and the number of participating MIoTDs is randomly chosen from [30, 54]. When an MIoTD is within a certain distance (d) from a micro-BS, it can communicate with the micro-BS directly. We assume the path loss models of micro-BSs and macro-BSs follow PL(dB) = 142.7 + 35.5 $\log_{10} d$, where d represents the distance between a micro-BS and its connected MIoTDs, as well as the distance between a micro-BS and the macro-BS, both measured in kilometers. The communication bandwidth of each micro-BS and each MIoTD node are 6 MHz and 2 MHz, respectively, which undergo Rayleigh fading. The noise power spectral density is $N_0 = -174$ dBm/Hz. Moreover, we consider that the size of tasks generated by each MIoTD obeys a uniform distribution between [200, 1200] KB, and the maximum computation capacities of an MIoTD and a macro-BS are 1 GHz and 25 GHz, respectively. The maximum computation capacity of each micro-BS is between 3 GHz to 7 GHz. The inverse of the power amplifier efficiency factor for the process of a micro-BS charging an MIoTD is $\eta = 1$. The energy-harvesting capability of the micro-BS is uniformly distributed between 0 and 80 J.

The DRL-based framework (DCTORA and DOS) employs neural networks in the simulations, specifically for each micro-BS agent. These neural networks are comprised of

an input layer, two hidden layers that are fully connected, and an output layer. In order to obtain the optimal objective value through the implementation of the gradient descent algorithm, it is necessary to periodically update the parameters of the target network and enhance the algorithm's convergence by employing experience replay memory. The actor network and the critic network have different learning rates of 0.0001 and 0.001, respectively, whereas the DOS utilizes a learning rate of 0.001. The neural network is composed of two hidden layers, including 200 and 100 neurons, respectively. Furthermore, the size of the experience replay buffer is set to 2000, the mini-batch size is set to 32, and the discount factor is set to 0.95. The maximum number of episodes is set to 1000, and the frequency of learning is set to 5. Table 3 provides a summary of the main parameter settings.

Table 3. Parameter values used in the simulations.

Parameters Value Range	Value
Number of macro-BS	1
Number of micro-BS	[5, 9]
Number of MIoTDs	[30, 54]
The communication bandwidth of each micro-BS	6 MHz
The communication bandwidth of each MIoTD node	2 MHz
Distance between MIoTD and micro-BS	[50, 200] m
Distance between MIoTD and macro-BS	[200, 1000] m
Noise power spectral density	−174 dBm/Hz
Task input size of each MIoTD in TS t	[200, 1200] KB
Energy harvesting capability of micro-BS m in TS t	[0, 80] J
The maximum latency of each task	[10, 80] ms
The maximum computation capacities of MIoTD, and macro-BS	1, 25 GHz
The maximum computation capacities of micro-BS	[3, 7] GHz
The maximum number of episodes	1000
The replay memory size	2000
The frequency of learning	5
The mini-batch size	32
The Adam optimizer's learning rate	0.01
The discount factor	0.95
The soft updating rate of target networks	0.001
The target network update frequency	50

5.2. Performance Analysis

We first evaluate the convergence of the proposed DCTORA algorithm. Following this, we analyze the impact of coefficient values on the cumulative reward, task latency, and carbon emissions. Moreover, we discuss the system performance of our solution in different scenarios by comparing it with benchmarks.

5.2.1. Convergence Analysis

Figure 3 shows the convergence performance of the proposed DCTORA algorithm with different values of w that are used in Equation (28). In the simulation, the number of participating MIoTDs is 30. Furthermore, the number of micro-BSs is 6 and its CPU frequency is 3 GHz. The energy-harvesting capability $G_m(t)$ of each micro-BS is set to 30 J. Then, we use our proposed Algorithm 1 to obtain the optimal task offloading strategy.

From Figure 3, we can observe that the cumulative reward converges to near-optimal values within 500 training episodes. Although the values of cumulative reward are in an unstable state with large fluctuations in the beginning, this indicates that each agent is constantly exploring the environment randomly. After a period of learning, the cumulative reward gradually stabilizes, and the fluctuation range decreases, regardless of the value of w. Therefore, we can conclude that our proposed algorithm has good convergence performance. In addition, it can be found that in Figure 3, when w decreases, the agent yields the highest efficacy of cumulative reward performance with $w = 0.2$, followed by $w = 0.5$ and then $w = 0.9$.

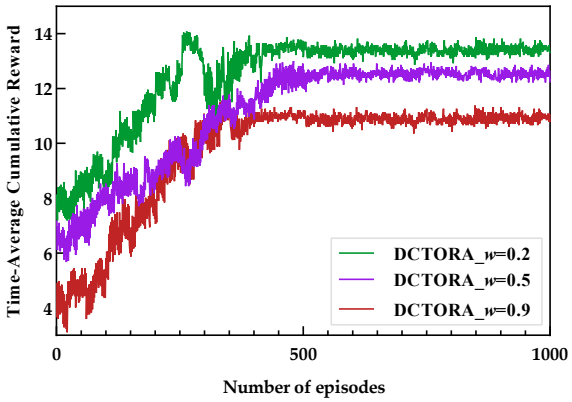

Figure 3. Cumulative reward of the proposed DCTORA algorithm under different values of coefficient, where $C_{m,n}(t) = 600$, $G_m(t) = 30$, $M = 6$, and $N = 30$.

5.2.2. Effect of Coefficient Value on Performance Metrics

To focus on the impact of w, we fix other parameters that are consistent with Figure 3. The results of the task latency and the carbon emission with different values of w are shown in Figure 4. As a whole, the values of task latency and carbon emission gradually increase initially and then converge to different stable values. As we can see from Figure 4a,b, the best outcome result in carbon emissions is attained when $w = 0.2$, followed by $w = 0.5$, and $w = 0.9$ has the lowest performance. The statistical results show that the carbon emission achieved with $w = 0.9$ surpasses that of $w = 0.5$ by a significant 35% and exceeds $w = 0.2$ by an impressive 38.24% across all 1000 training rounds. The task latency, when $w = 0.2$, shows an increase of 27.14% compared to $w = 0.5$ and a substantial 18.27% increase compared to $w = 0.9$. From (28) and (29), we can directly observe that when $w < 0.5$, more emphasis is placed on the effect of carbon emission. Conversely, when $w > 0.5$, more emphasis is placed on the impact of task latency. To balance the trade-off between task latency and carbon emissions, we choose $w = 0.5$ for the rest of the simulations in which the value of w is not the variable.

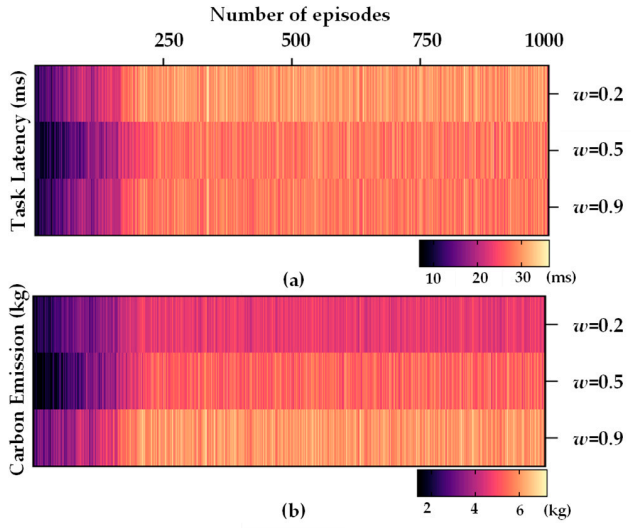

Figure 4. Average system performance metrics of DCTORA: (**a**) task latency, (**b**) carbon emission, where $C_{m,n}(t) = 600$, $G_m(t) = 30$, $M = 6$, and $N = 30$.

5.2.3. Performance Metrics Versus Varying Data Size of MIoTD

Figure 5 compares the performance of different offloading schemes in terms of execution efficiency, task latency, and carbon emissions under different values of $C_{m,n}(t)$, where $w = 0.5$, $G_m(t) = 40$, $f_m^{\max} = 4$, $p_{m,n}^{\max} = 15$, and $M = 6$. In Figure 5a, as $C_{m,n}(t)$ increases, the

values of execution efficiency gradually decrease. The execution efficiency is composed of the task latency and carbon emissions; a larger $C_{m,n}(t)$ implies tasks with larger sizes arriving in each TS, which usually leads to heavier computation and transmission burdens for task offloading and results in larger task latency and carbon emissions, subject to the limited computing capacity of MIoTDs. Meanwhile, our proposal achieves better execution efficiency performance than the others in each case (at least 50% larger than FL), and the DDPG-based agents perform better than the DQN-based ones. The GP method seeks to complete more tasks and consumes too much traditional energy, which makes short-sighted choices based on the task latency but does not consider the carbon emission. The FO scheme does not employ local computing, so there is a gap in GP. Furthermore, the FL algorithm overly relies on local execution capabilities, leading to its long latency. To gain a more insightful understanding, we plot the task latency and carbon emission with different values of $C_{m,n}(t)$ in Figure 5b,c.

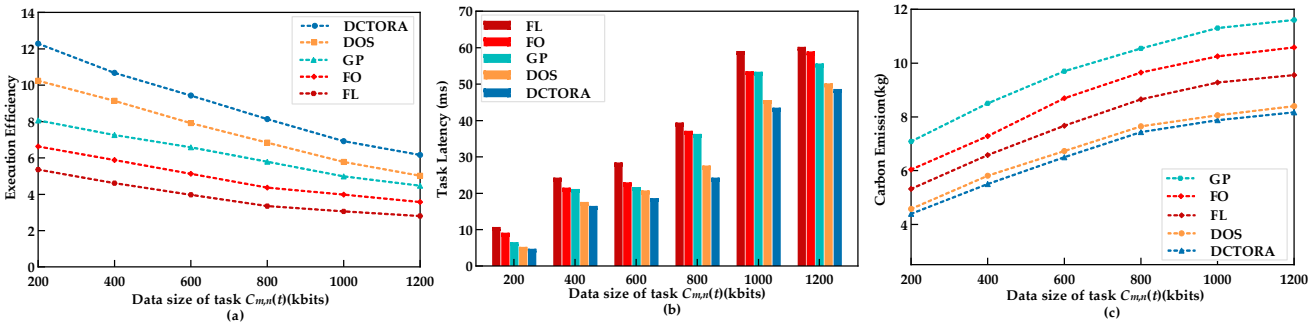

Figure 5. (**a**) Execution efficiency, (**b**) task latency, and (**c**) carbon emissions versus different sizes of tasks under different schemes, where $w = 0.5$, $G_m(t) = 40$, $f_m^{\max} = 4$, $p_{m,n}^{\max} = 15$, and $M = 6$.

In Figure 5b,c, the trend of the task latency and carbon emissions both go up, with $C_{m,n}(t)$ becoming larger and larger. In Figure 5b, the difference in task latency among the five policies is slight when $C_{m,n}(t) < 600$. The DCTORA exhibits the smallest delay, while the FL has the largest delay. The GP's task latency is less than the FO and FL algorithms. This is because the FL algorithm has the lowest computing capacity, resulting in its task latency being the longest. The GP algorithm seeks to complete more tasks locally or remotely; however, it ignores energy consumption. In contrast to GP, FO lacks local executions, so its task latency is slightly longer than GP. In Figure 5c, we can observe a significant increase in carbon emissions as $C_{m,n}(t)$ increases. DCTORA exhibits the smallest value, surpassing the GP algorithm by 18%, followed by DOS, FL, FO, and GP. The main reason is that as $C_{m,n}(t)$ rises, more tasks are required to be executed either locally or transferred to the micro- or macro-BS. This leads to higher power consumption, potentially resulting in renewable energy sources failing to meet the system's energy demand. Consequently, additional grid energy is consumed, leading to increased carbon emissions. The FL algorithm has lower local computing energy consumption for MIoTD compared to the server, resulting in reduced carbon emissions compared to the FO and GP algorithms. Although the FL algorithm may contribute to a carbon emission reduction, it ignores the issue of task latency.

Furthermore, we can conclude from the above that two DRL-based methods (DOS and DDPG) outperform other benchmarks in terms of execution efficiency, task latency, and carbon emission performances. In each $C_{m,n}(t)$ case, the DCTORA scheme exhibits a reduced task delay and carbon emissions compared to the DOS, providing additional evidence of its effectiveness in addressing high-dimensional complex problems involving continuous action-state spaces.

5.2.4. Performance Metrics Versus Varying Energy-Harvesting Capability of Micro-BS

Figure 6 shows the performance comparison in terms of execution efficiency, task latency, and carbon emissions under different energy-harvesting capabilities of micro-BS, where $w = 0.5$, $C_{m,n}(t) = 600$, $f_m^{\max} = 4$, $p_{m,n}^{\max} = 15$, and $M = 6$. As we can see from Figure 6a,

the execution efficiency gradually increases with an increase in $G_m(t)$. This is because, with a higher value of $G_m(t)$, the system has more renewable energy and prefers to process more tasks locally or offload them to the edge servers. As $G_m(t) > 60$, the harvested renewable energy in real-time is sufficient to support the requirement of task executions, execution efficiency remains relatively constant regardless of the increase in the capability of energy harvesting. In Figure 6b,c, the task latency and carbon emission gradually decrease and then also converge to stable values with the increase in the $G_m(t)$. The reason is that, as $G_m(t)$ increases, more green energy becomes available for the system, reducing grid energy consumption and decreasing carbon emissions. Additionally, more available green energy enables local and remote task execution and powers more computations, leading to reduced task latency and, as a result, improved execution efficiency.

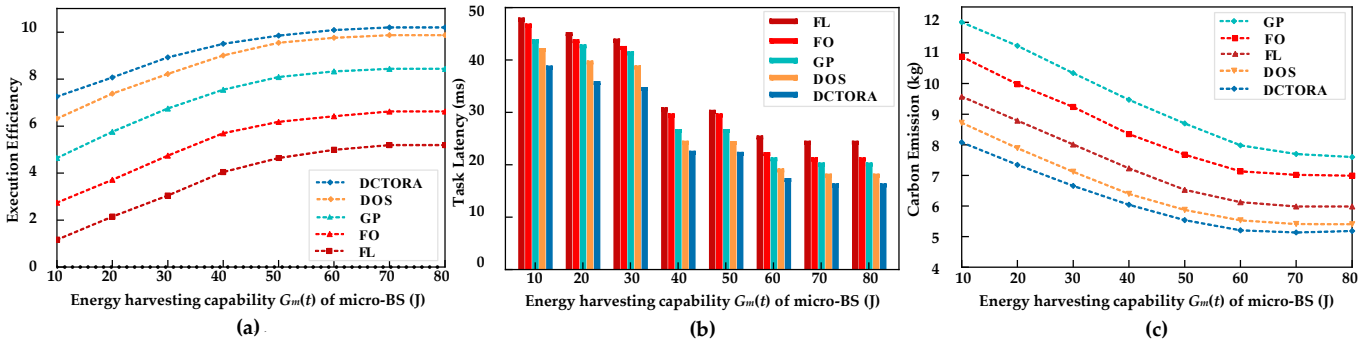

Figure 6. (**a**) Execution efficiency, (**b**) task latency, and (**c**) carbon emission versus different energy-harvesting capabilities $G_m(t)$ under different schemes, where $w = 0.5$, $C_{m,n}(t) = 600$, $f_m^{\max} = 4$, $p_{m,n}^{\max} = 15$, and $M = 6$.

As a whole, we can demonstrate that the DRL-based algorithms can make proper offloading decisions to achieve better performance, and the simulation results show that the performance of the DCTORA algorithm is higher than the other schemes. As for the traditional algorithms, the GP's result is better than the other methods, but it only makes short-sighted choices based on the task latency, not considering the carbon emissions. When the value of $G_m(t)$ increases, all algorithms' execution efficiencies increase, and task latency and carbon emission decrease (compared with the GP algorithm, the DCTORA algorithm has changed more than 26% and 18% in task latency and carbon emissions, respectively).

5.2.5. Performance Metrics Versus Maximum Transmit Power of MIoTD

The maximum transmission power of MIoTD has a significant impact on the data transmission delay, energy consumption, carbon emissions, and the number of data bits executed by the system. Hence, this section will delve into exploring how $p_{m,n}^{\max}$ influences the system's performance. Figure 7 shows the correlation between the system performance and $p_{m,n}^{\max}$ of MIoTDs by different offloading policies, where $w = 0.5$, $G_m(t) = 50$, $C_{m,n}(t) = 600$, $f_m^{\max} = 4$, and $M = 6$. Figure 7a illustrates the execution efficiency under different $p_{m,n}^{\max}$ constraints at the MIoTD for different offloading schemes. The curves show that the execution efficiency of different algorithms except FL increases as $p_{m,n}^{\max}$ increases. The increase in $p_{m,n}^{\max}$ indicates a high available transmission rate between MIoTD and the micro-BS, which can facilitate task offloading and increase energy consumption to a certain extent, while also leading to increased carbon emissions when renewable energy is Insufficient. The DCTORA scheme achieves better performance than other offloading methods under different $p_{m,n}^{\max}$ constraints, which indicates its effectiveness.

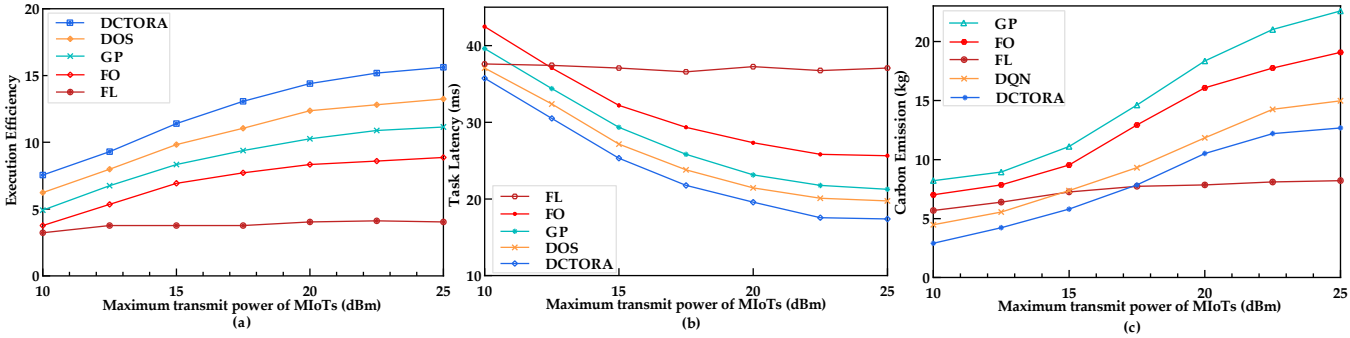

Figure 7. (**a**) Execution efficiency, (**b**) task latency, and (**c**) carbon emission versus different maximum transmit powers of MIoTDs under different schemes, where $w = 0.5$, $G_m(t) = 50$, $C_{m,n}(t) = 600$, $f_m^{\max} = 4$, and $M = 6$.

We can see from Figure 7b,c that the FL algorithm is almost not affected by the change in $p_{m,n}^{\max}$, other schemes are highly dependent on the $p_{m,n}^{\max}$ constraint, the task latency decreases significantly as $p_{m,n}^{\max}$ increases, and the carbon emissions show a rising trend with an increase in $p_{m,n}^{\max}$. This is because in FL, each MIoTD executes tasks locally without sending them to micro-BSs. For other schemes, higher levels of power allocated to MIoTDs can directly improve MIoTDs' transmission rates, and the MIoTDs will prefer to offload tasks to the high-performance processor within their competence, which can facilitate task offloading and increase carbon emissions to a certain extent. According to Figure 7b, the FL, FO, and GP algorithms are ineffective at minimizing task delay, with increases of 35.9%, 25.5%, and 10.1%, respectively, as compared to the proposed algorithm. Meanwhile, the simulation results showed that the DOS scheme had a task latency that was 4.9% greater than that of the proposed algorithm.

5.2.6. Performance Metrics Versus Different Computation Capacity of Micro-BS

Figure 8 presents the influence of different values of $f_m^{\max}$ (ranging from 3 GHz to 7 GHz) on system performance under different schemes with a fixed number of micro-BSs, where we set $G_m(t) = 20$, $C_{m,n}(t) = 600$, $p_{m,n}^{\max} = 13$, and $M = 6$. As we can see from Figure 8a, the execution efficiency values of all curves increase when the computation capacity of the micro-BS increases. This is because a larger $f_m^{\max}$ means more computing capability can be provided, and it will considerably decrease the task execution latency at the micro-BS. When the computation capacity of the micro-BS reaches up to 5, the execution efficiency of all algorithms increases slowly.

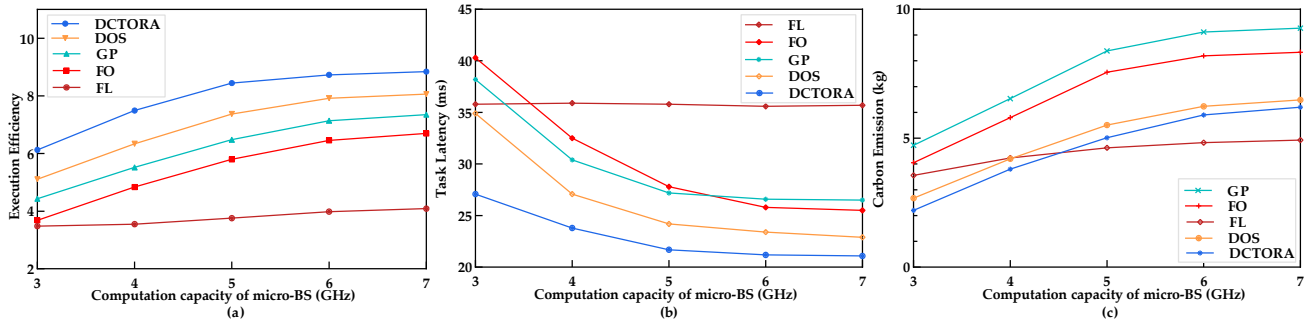

Figure 8. (**a**) Execution efficiency, (**b**) task latency, and (**c**) carbon emission versus different computation capacities of micro-BS under different schemes, where $w = 0.5$, $G_m(t) = 20$, $C_{m,n}(t) = 600$, $p_{m,n}^{\max} = 13$, and $M = 6$.

The effect of $f_m^{\max}$ on task latency and carbon emissions is shown in Figure 8b,c, where we fix $f_{m,n}^{\max}$ and increase $f_m^{\max}$ of the micro-BS. We can observe that the task latency and carbon emissions of the FL algorithm are less affected by $f_m^{\max}$. Under different values

of $f_m^{\max}$, the DOS and DCTORA schemes can achieve better performance than the other three benchmarks, and the performance of the DCTORA paradigm is slightly better than the DOS paradigm. When $f_m^{\max} < 5$ GHz/s, except FL, the task latency of all algorithms obviously decreases. This is because the processing speed of the micro-BS is faster than that of the MIoTD, and each MIoTD chooses to offload more tasks to the micro-BS. When 5 GHz/s $\leq f_m^{\max} \leq 7$ GHz/s, the downward trend of task latency slows down. Numerically, DCTORA is 18.4%, 38.9%, 23.5%, and 21.2% better than DOS, FL, FO, and GP in task latency, respectively. From Figure 8c, we can observe that the carbon emissions of all algorithms, except FL, increase with an increase 0k $f_m^{\max}$ when 5 GHz/s $\leq f_m^{\max} \leq 6$ GHz/s and gradually stabilizes at $f_m^{\max} > 6$ GHz/s. When $f_m^{\max} > 6$ GHz/s, the system's energy consumption has reached saturation, even if increasing $f_m^{\max}$ has little influence on the system's carbon emissions. Furthermore, the performance of DCTORA is superior to other algorithms. The GP algorithm seeks to accomplish more tasks, so it has the highest carbon emissions. FO and FL algorithms take into account the carbon emissions factor, but the FL algorithm lacks BS-aided computing with traditional energy supply, so it has slightly lower carbon emissions. To summarize, DCTORA is 8.9%, 6.1%, 29.9%, and 31.9% lower than DOS, FL, FO, and GP in carbon emissions, respectively.

5.2.7. Performance Metrics Versus Different Numbers of Micro-BSs

To gain a more insightful understanding of the DCTORA agent, we plot the execution efficiency, task latency, and carbon emission values with different numbers of micro-BSs. We take a random number between [30, 54] as the total number of MIoTDs and set $f_m^{\max} = 4$ GHz/sec, $G_m(t) = 20$, and $C_{m,n}(t) = 600$. Taken as a whole, except for FL, the other algorithms' execution efficiencies increase, task latency decreases, and carbon emissions increase as the number of micro-BSs grows, as shown in Figure 9. Our proposed DCTORA algorithm can achieve the best result, and the DOS follows with a small gap. The reason for this is that increased micro-BS availability offers more computational resources, and MIoTDs are closer to the micro-BS, which allows more MIoTDs to transfer their tasks to neighboring micro-BSs for processing, resulting in lower transmission and execution latency and improved system execution efficiency but increasing carbon emissions to some extent. The performance of FL remains unaffected by the increasing number of micro-BSs, mainly because local computing does not employ the computational resources offered by micro-BSs' high-performance processors.

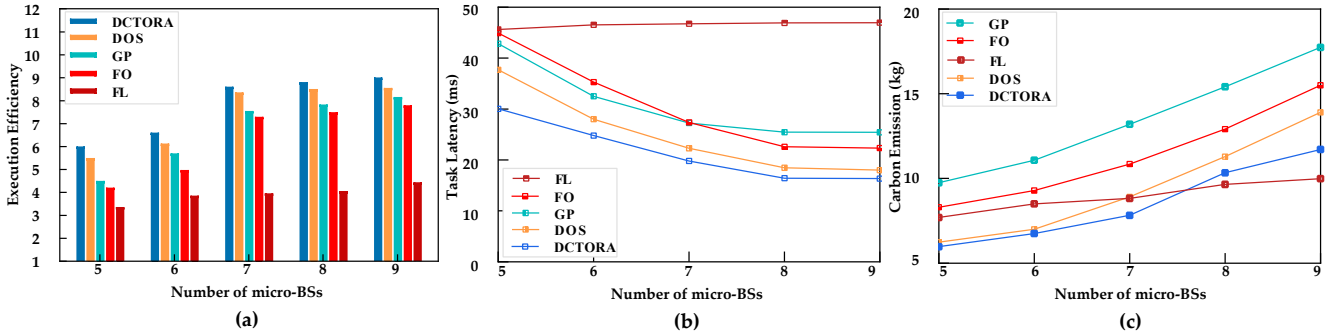

Figure 9. (**a**) Execution efficiency, (**b**) task latency, and (**c**) carbon emissions versus different numbers of micro-BSs under different schemes, where $w = 0.5$, $G_m(t) = 20$, $C_{m,n}(t) = 600$, $f_m^{\max} = 4$, and $p_{m,n}^{\max} = 15$.

6. Conclusions

In this paper, we investigate the joint optimization of task offloading and resource allocation in MEC-enabled maritime IoT networks. We propose a framework for green MEC-enabled MIoT networks and present the total system execution efficiency as the system's performance metric that covers the total size of completed tasks, task execution latency, and the system's carbon emissions. The joint task offloading and resource allocation problem is

formulated by optimizing the total system execution efficiency, and a DDPG-based joint optimization strategy is proposed to solve the problem. By interacting with the time-varying wireless channel conditions, randomly arriving renewable energy and computing tasks, and continuous action space, our algorithm can learn the optimal strategy to significantly reduce the system's carbon emissions and task delay and improve the total size of completed tasks. The simulation results verify the superior performance of our algorithm. In the future, we will delve into other novel methods, such as federated learning, model pruning, and knowledge distillation, to effectively minimize the computational burden on models, further improve the utilization of renewable energy, and reduce greenhouse gas emissions.

Author Contributions: Conceptualization, Z.W., R.H. and C.S.; methodology, Z.W. and R.H.; software, Z.W. and Y.L.; formal analysis, Z.W. and R.H.; investigation, Z.W. and C.S.; writing—original draft preparation, Z.W. and R.H.; writing—review and editing, Z.W., R.H., Y.L. and C.S.; visualization, Z.W.; supervision, R.H.; project administration, R.H.; funding acquisition, R.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially supported by the National Natural Science Foundation of China (61371091, 61801074, and 62371085) and the Dalian Science and Technology Innovation Fund (2019J11CY015).

Institutional Review Board Statement: No applicable.

Data Availability Statement: Data are contained within the article.

Acknowledgments: The authors would like to thank the editors and reviewers for their comments on the manuscript of this article.

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

References

1. Qiu, T.; Zhao, Z.; Zhang, T.; Chen, C.; Chen, C.P. Underwater Internet of Things in smart ocean: System architecture and open issues. *IEEE Trans. Ind. Inform.* **2019**, *16*, 4297–4307. [CrossRef]
2. Nomikos, N.; Gkonis, P.K.; Bithas, P.S.; Trakadas, P. A Survey on UAV-Aided Maritime Communications: Deployment Considerations, Applications, and Future Challenges. *IEEE Open J. Commun. Soc.* **2023**, *4*, 56–78. [CrossRef]
3. Li, K.; Wang, X.; Ni, Q.; Huang, M. Entropy-based Reinforcement Learning for computation offloading service in software-defined multi-access edge computing. *Future Gener. Comp. Syst.* **2022**, *136*, 241–251. [CrossRef]
4. Lin, Z.; Chen, X.; Chen, P. Energy harvesting space-air-sea integrated networks for MEC-enabled maritime Internet of Things. *China Commun.* **2022**, *19*, 47–57. [CrossRef]
5. Giannopoulos, A.; Nomikos, N.; Ntroulias, G.; Syriopoulos, T.; Trakadas, P. Maritime Federated Learning for Decentralized On-Ship Intelligence. In Proceedings of the International Conference on Artificial Intelligence Applications and Innovations, León, Spain, 14–17 June 2023.
6. Jang, J.; Tulkinbekov, K.; Kim, D.-H. Task Offloading of Deep Learning Services for Autonomous Driving in Mobile Edge Computing. *Electronics* **2023**, *12*, 3223. [CrossRef]
7. Xiao, G.; Zhang, H.; Hassan, H.; Chen, Y.; Huang, Z.; Sun, N. A cooperative offloading game on data recovery for reliable broadcast in VANET. *Concurr. Comput.-Pract. Exp.* **2017**, *29*, e3938. [CrossRef]
8. Mao, Y.; Zhang, J.; Letaief, K.B. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 3590–3605. [CrossRef]
9. Barbarossa, S.; Sardellitti, S.; Di Lorenzo, P. Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks. *IEEE Signal Proc. Mag.* **2014**, *31*, 45–55. [CrossRef]
10. Zhang, W.; Wen, Y.; Guan, K.; Kilper, D.; Luo, H.; Wu, D.O. Energy-optimal mobile cloud computing under stochastic wireless channel. *IEEE Trans. Wirel. Commun.* **2013**, *12*, 4569–4581. [CrossRef]
11. Borkar, V.S.; Choudhary, S.; Gupta, V.K.; Kasbekar, G.S. Scheduling in wireless networks with spatial reuse of spectrum as restless bandits. *Perform. Eval.* **2021**, *149–150*, 102208. [CrossRef]
12. Niu, Z. TANGO: Traffic-aware network planning and green operation. *IEEE Wirel. Commun.* **2011**, *18*, 25–29. [CrossRef]
13. Wang, J.; Ge, Y. A radio frequency energy harvesting-based multihop clustering routing protocol for cognitive radio sensor networks. *IEEE Sens. J.* **2022**, *22*, 7142–7156. [CrossRef]
14. Hu, H.; Da, X.; Ni, L.; Huang, Y.; Zhang, H. Green energy powered cognitive sensor network with cooperative sensing. *IEEE Access* **2019**, *7*, 17354–17364. [CrossRef]
15. Sun, Y.; He, Q. Computational Offloading for MEC Networks with Energy Harvesting: A Hierarchical Multi-Agent Reinforcement Learning Approach. *Electronics* **2023**, *12*, 1304. [CrossRef]

16. Pasha, M.; Rahman Khan, K.U. Scalable and energy efficient task offloading schemes for vehicular cloud computing. *Int. J. Comput. Netw. Commun.* **2018**, *10*, 35–52. [CrossRef]
17. Zhang, G.; Zhang, W.; Cao, Y.; Li, D.; Wang, L. Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4642–4655. [CrossRef]
18. Zhang, Y.; He, J.; Guo, S. Energy-Efficient Dynamic Task Offloading for Energy Harvesting Mobile Cloud Computing. In Proceedings of the 2018 IEEE International Conference on Networking, Architecture and Storage (NAS), Chongqing, China, 11–14 October 2018.
19. 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.
20. Giannopoulos, A.; Spantideas, S.; Capsalis, N.; Gkonis, P.; Karkazis, P.; Sarakis, L.; Trakadas, P.; Capsalis, C. WIP: Demand-Driven Power Allocation in Wireless Networks with Deep Q-Learning. In Proceedings of the 2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), Pisa, Italy, 7–11 June 2021.
21. Xiong, X.; Zheng, K.; Lei, L.; Hou, L. Resource allocation based on deep reinforcement learning in IoT edge computing. *IEEE J. Sel. Areas Commun.* **2020**, *38*, 1133–1146. [CrossRef]
22. Ma, H.; Huang, P.; Zhou, Z.; Zhang, X.; Chen, X. GreenEdge: Joint green energy scheduling and dynamic task offloading in multi-tier edge computing systems. *IEEE Trans. Veh. Technol.* **2022**, *71*, 4322–4335. [CrossRef]
23. Vamvoudakis, K.G.; Lewis, F.L. Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica* **2010**, *46*, 878–888. [CrossRef]
24. Wei, Y.; Yu, F.R.; Song, M.; Han, Z. User scheduling and resource allocation in HetNets with hybrid energy supply: An actor-critic reinforcement learning approach. *IEEE Trans. Wirel. Commun.* **2017**, *17*, 680–692. [CrossRef]
25. Wei, Z.; He, R.; Li, Y. Deep Reinforcement Learning Based Task Offloading and Resource Allocation for MEC-Enabled IoT Networks. In Proceedings of the 2023 IEEE/CIC International Conference on Communications in China (ICCC Workshops), Dalian, China, 10–12 August 2023.
26. Fang, X.; Feng, W.; Wang, Y.; Chen, Y.; Ge, N.; Ding, Z. NOMA-Based Hybrid Satellite-UAV-Terrestrial Networks for Beyond 5G Maritime Internet of Things. *IEEE Trans. Wirel. Commun.* **2021**, *22*, 138–152. [CrossRef]
27. Anbalagan, S.; Kumar, D.; Raja, G.; Balaji, A. SDN assisted Stackelberg Game model for LTE-WiFi offloading in 5G networks. *Digit. Commun. Netw.* **2019**, *5*, 268–275. [CrossRef]
28. El Haber, E.; Nguyen, T.M.; Assi, C. Joint optimization of computational cost and devices energy for task offloading in multi-tier edge-clouds. *IEEE Trans. Commun.* **2019**, *67*, 3407–3421. [CrossRef]
29. Ning, Z.; Dong, P.; Kong, X.; Xia, F. A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things. *IEEE Internet Things* **2018**, *6*, 4804–4814. [CrossRef]
30. Jiang, F.; Wei, F.; Wang, J.; Liu, X. Delay-Aware Energy Minimization Offloading Scheme for Mobile Edge Computing. In Proceedings of the 2020 IEEE/CIC International Conference on Communications in China (ICCC), Chongqing, China, 9–11 August 2020.
31. Hu, X.; Wang, L.; Wong, K.K.; Tao, M.; Zhang, Y.; Zheng, Z. Edge and central cloud computing: A perfect pairing for high energy efficiency and low-latency. *IEEE Trans. Wirel. Commun.* **2019**, *19*, 1070–1083. [CrossRef]
32. Ma, X.; Zhao, J.; Li, Q.; Gong, Y. Reinforcement Learning Based Task Offloading and Take-Back in Vehicle Platoon Networks. In Proceedings of the 2019 IEEE International Conference on Communications Workshops (ICC Workshops), Shanghai, China, 10–24 May 2019.
33. Xu, S.; Liu, Q.; Gong, B.; Qi, F.; Guo, S.; Qiu, X.; Yang, C. RJCC: Reinforcement-learning-based joint communicational-and-computational resource allocation mechanism for smart city IoT. *IEEE Internet Things* **2020**, *7*, 8059–8076. [CrossRef]
34. Kiran, N.; Pan, C.; Wang, S.; Yin, C. Joint resource allocation and computation offloading in mobile edge computing for SDN based wireless networks. *J. Commun. Netw.* **2019**, *22*, 1–11. [CrossRef]
35. Li, Y.; Wang, T.; Wu, Y.; Jia, W. Optimal dynamic spectrum allocation-assisted latency minimization for multiuser mobile edge computing. *Digit. Commun. Netw.* **2022**, *8*, 247–256. [CrossRef]
36. Jiang, F.; Wang, K.; Dong, L.; Pan, C.; Xu, W.; Yang, K. Deep-learning-based joint resource scheduling algorithms for hybrid MEC networks. *IEEE Internet Things* **2019**, *7*, 6252–6265. [CrossRef]
37. Wang, R.; Jiang, X.; Zhou, Y.; Li, Z.; Wu, D.; Tang, T.; Fedotov, A.; Badenko, V. Multi-agent reinforcement learning for edge information sharing in vehicular networks. *Digit. Commun. Netw.* **2022**, *8*, 267–277. [CrossRef]
38. Yang, T.; Feng, H.; Gao, S.; Jiang, Z.; Qin, M.; Cheng, N.; Bai, L. Two-stage offloading optimization for energy-latency tradeoff with mobile edge computing in maritime Internet of Things. *IEEE Internet Things* **2019**, *7*, 5954–5963. [CrossRef]
39. Su, X.; Xue, H.; Zhou, Y.; Zhu, J. Research on computing offloading method for maritime observation monitoring sensor network. *J. Commun.* **2021**, *42*, 149–163. [CrossRef]
40. Su, X.; Wang, Z.Y.; Wang, Y.P.; Zhou, S.Y. Multi-access edge computing offloading in maritime monitoring sensor networks. *Chin. J. Internet Things* **2021**, *5*, 36–52. [CrossRef]
41. Nath, S.; Wu, J. Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems. *Intell. Converg. Net.* **2020**, *1*, 181–198. [CrossRef]
42. Chen, J.; Xing, H.; Xiao, Z.; Xu, L.; Tao, T. A DRL agent for jointly optimizing computation offloading and resource allocation in MEC. *IEEE Internet Things* **2021**, *8*, 17508–17524. [CrossRef]

43. Rabaey, J.M. *Digital Integrated Circuits a Design Perspective*; Prentice Hall: Upper Saddle River, NJ, USA, 1999.
44. Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.
45. Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; Freitas, N. Dueling network architectures for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016.
46. Cao, S.; Chen, S.; Chen, H.; Zhang, H.; Zhan, Z.; Zhang, W. HCOME: Research on Hybrid Computation Offloading Strategy for MEC Based on DDPG. *Electronics* **2023**, *12*, 562. [CrossRef]
47. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjel, A.K.; Ostrovski, O.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
48. Jo, S.; Kim, U.; Kim, J.; Jong, C.; Pak, C. Deep reinforcement learning-based joint optimization of computation offloading and resource allocation in F-RAN. *IET Commun.* **2023**, *17*, 549–564. [CrossRef]