Multi-agent reinforcement learning for cost-aware collaborative task execution in energy-harvesting D2D networks

Binbin Huang[a], Xiao Liu[b], Shangguang Wang[c],
Linxuan Pan[d], Victor Chang[e,*]

[a]*School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China,310018*

[b]*School of Information Technology, Deakin University, Australia*

[c]*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China*

[d]*State Key Laboratory for Novel Software Technology, Software Institute, Nanjing University, Nanjing, China*

[e]*Artificial Intelligence and Information Systems Research Group, School of Computing, Engineering and Digital Technologies, Teesside University, Middlesbrough, UK*

Abstract In device-to-device (D2D) networks, multiple resource-limited mobile devices cooperate with one another to execute computation tasks. As the battery capacity of mobile devices is limited, the computation tasks running on the mobile devices will terminate once the battery is dead. In order to achieve sustainable computation, energy-harvesting technology has been introduced into D2D networks. At present, how to make multiple energy harvesting mobile devices work collaboratively to minimize the long-term system cost for task execution under limited computing, network and battery capacity constraint is a challenging issue. To deal with such a challenge, in this paper, we design a multi-agent deep deterministic policy gradient (MADDPG) based cost-aware collaborative task-execution (CACTE) scheme in energy harvesting D2D (EH-D2D) networks. To validate the CACTE scheme's performance, we conducted extensive experiments to compare the CACTE scheme with four baseline algorithms, including Local, Random, ECLB (Energy Capacity Load Balance) and CCLB (Computing Capacity Load Balance). Experiments were accompanied by various system parameters, such as the mobile device's battery capacity, task workload, the bandwidth and so on. The experimental results show that the CACTE scheme can make multiple mobile devices cooperate effectively with one another to execute many more tasks and achieve a higher long-term reward, including lower task latency and fewer dropped tasks.

Keywords: D2D networks, collaborative task execution, cost-aware, partially observable Markov decision process, multi-agent deep deterministic policy gradient

1. Introduction

Mobile devices are typically constrained in their computation, network and battery resources[1][2]. Device-to-device (D2D) communication technology enables two physically proximate mobile devices to directly communicate with each other [3]-[6]. In D2D communication networks, computation tasks can be collaboratively executed by offloading them from mobile devices with insufficient resources to those with abundant resources to take full advantage of the available resources, called collaborative task execution. Existing works mainly focus on the problem of collaborative task execution with limited computation resources [7]-[13]. However, the limited battery capacity of mobile devices still poses another critical challenge. For example, computation tasks running on a battery-powered mobile device will terminate once the battery is dead, which significantly impairs the quality of mobile services.

The energy-harvesting technique has been introduced to empower energy-constrained mobile devices and promote better performance. By accessing harvested energy, the computation capacity of mobile devices can be enhanced to achieve sustainable collaborative task execution. The sustainable collaborative task execution enables the computation tasks to be processed in an edge network, which helps reduce the service latency and improve the service quality and avoids possible performance bottlenecks caused by offloading a large number of tasks to edge servers. However, integrating energy harvesting into D2D networks also introduces new challenges to collaborative task execution. For example, due to the unpredictable amount of harvested energy, it is hard to determine collaborative task-execution solutions (e.g., the number of locally executed tasks, the number of offloaded tasks and the maximum number of received tasks) for mobile devices.

Existing studies focus on collaborative task execution either with battery-powered devices in conventional D2D networks or with edge servers in wireless-powered mobile edge computing (MEC). However, there exist a few major problems:

(1) Most studies have failed to consider that the battery in mobile devices is limited. For example, the authors in [14] and [15] mainly focus on solving the optimal task offloading schemes to optimize the energy consumption of systems but without considering the battery constraints; hence, these schemes are not suitable for solving the problem of collaborative task execution in energy harvesting D2D (EH-D2D) networks.

(2) Most studies have failed to consider the unique requirements for collaborative task execution by multiple energy-harvesting mobile devices in D2D networks. For example, the authors in [16] [17] [18] and [19] mainly investigate that the scenario where tasks are collaboratively executed by edge servers in wireless powered MEC. However, there are many differences between the collaborative task execution in wireless powered MEC and EH-D2D networks. Firstly, in comparison with an edge server, the coverage area of mobile devices is much smaller. Secondly, the computing, communication and battery capacities of mobile devices are much lower than those of edge servers. Thirdly, the mobility of mobile devices makes collaborative task execution in D2D networks much more intractable compared to the MEC environment.

(3) Most studies have failed to consider the long-term cost (e.g., the weighted sum of the average latency of task processing, the number of dropped tasks and the battery energy penalty) for the collaborative task execution.

This paper aims to address the above problems, with a particular focus on minimizing long-term system costs. We firstly present a system model, including EH-D2D system architecture, energy queueing and so on. We then formulate the multi-user collaborative task-execution problem as a partially observable Markov decision process (POMDP), where each mobile device is treated as an agent. The observation space for each agent in our model mainly includes the following: the execution queue's state; the channel gain between mobile devices; the available battery energy of the mobile device; the channel gain between mobile devices and the power beacon; and the number of arrival tasks. In this model, each mobile device makes an optimal collaborative task-execution policy to minimize the long-term system cost based on its observable space. In this paper, we also describe our design for a cost-aware collaborative task-execution scheme (CACTE) based on a multi-agent deep deterministic policy gradient (MADDPG). Finally, in the total reward, the average latency of tasks processing, the number of dropped tasks and the battery energy penalty different performance metrics. We discuss the result of comprehensive experiments that were conducted to evaluate and compare the CACTE scheme with four baseline algorithms, including Local, Random, ECLB (Energy Capacity Load Balance) and CCLB

(Computing Capacity Load Balance). The experimental results show that our proposed CACTE scheme achieves the highest total reward, the lowest average latency and the smallest number of dropped tasks.

The preliminary work for this paper was accepted by *HPCC* in 2020. The major extension presented here includes model construction, the experiments conducted and discussion. In terms of the model's construction, we further consider the influence of uplink and downlink transmission rates on collaborative task execution. Moreover, we add the uplink and downlink transmission rates to the state space, and we consider the effects of transmission rate constraints on executable action. As for the experiments, two new baseline algorithms were added, and we compared our proposed algorithm with baseline algorithms, using different experimental parameters, such as task arrival rate, battery capacity, task workload, bandwidth and the number of mobile users. Finally, we now present further discussion of the related experimental results to validate our study.

Here, we summarize the main contributions of our work, as follows:

- We mainly focus on how to coordinate multiple energy-harvesting mobile devices with limited battery capacity to execute computation tasks in EH-D2D networks.
- We formulate the multi-user collaborative task-execution problem as a POMDP. Our goal is to minimize the long-term system cost, which is defined as the weighted sum of the average latency of task processing, the number of dropped tasks and the battery energy penalty.
- We propose a MADDPG-based CACTE scheme to solve this problem. Meanwhile, to validate the effectiveness of our proposed CACTE scheme, comprehensive experiments with different experimental parameters (such as battery capacity, bandwidth, number of mobile users and so on) have been conducted.

The remainder of this paper is organized as follows: Section 2 reviews the related works. Section 3 presents the system model. In Section 4, the multi-user collaborative task-execution problem is formulated. Section 5 describes the details of the CACTE scheme (based on the MADDPG). Section 6 presents the simulation experiments and analyzes the experimental results. Finally, Section 7 concludes this paper.

2. Related works

Many studies on conventional battery-powered D2D networks have been dedicated to studying the collaborative task-execution problem from different optimization objectives. For example, in [14], a Lyapunov-based online task-offloading algorithm is designed to solve the task offloading problem in D2D networks to minimize the time-average energy consumption for the task executions of all users. In [20], an auction-based incentive mechanism is proposed to solve the collaborative task-offloading problem. The objective is to optimize the long-term system welfare without knowledge of future information. In [21], a graph-matching-based optimal task-assignment policy is proposed to solve the collaborative task-execution problem and thereby significantly reduce the mobile devices' energy consumption. In [22], an alternating descent algorithm is proposed to solve the joint allocation problem of D2D networks, bandwidth, and power in the cognitive unmanned aerial vehicle (UAV)-enabled networks to support the ground terminals (GTs) while guaranteeing the quality-of-service for the D2D users. In [23], a non-orthogonal multiple-access (NOMA)-aided computing scheme is proposed to solve the task-offloading problem. The objective is to minimize energy consumption and maximize offloading data. However, it is realistic that the mobile device's battery energy is limited. All the works mentioned above ignore the impact of limited battery energy on collaborative task execution. For example, when the mobile device's battery energy is low, it will no longer accept offloaded tasks from other mobile devices, and it may offload its own tasks to other resource-rich mobile devices in the meantime. In the worse situation, tasks running on a battery-powered mobile device will be terminated when the battery is dead, which greatly impairs application performance.

In order to enable sustainable computing for battery-constrained mobile devices, energy-harvesting technology has been introduced. For example, in [16], an online offloading framework based on deep reinforcement learning (DRL) is designed to learn an optimal task-offloading and wireless resource allocation scheme in wireless powered MEC to achieve near-optimal performance while significantly decreasing the computation time. In [17], two DRL-based algorithms are proposed to learn the optimal server selection, offloading ratio and local computation policy in MEC systems with energy-harvesting devices, the goal of which is to balance the time and energy consumed. In [18], an efficient cooperation method is proposed to optimize the amount of offloaded task data, the system time allocation, and the transmit power in wireless powered MEC systems to maximize the amount of processed data. In [24], the Lyapunov optimization theory is utilized to optimize the computation offloading strategy, transmission power and so on in the D2D-aided wireless powered MEC system, the goal of which is to maximize the long-term utility energy efficiency. In [25], convex optimization techniques are adopted to jointly optimize transmission energy allocation and task allocation to minimize total transmission energy consumption. However, all the above studies mainly investigate the collaborative task-execution problem between energy-harvesting mobile devices and edge servers in wireless powered MEC. They cannot be readily applied to EH-D2D networks. It is much more intractable to solve the collaborative task-execution problem subject to the energy constraints of mobile device batteries in EH-D2D networks compared with wireless powered MEC.

Motivated by this, we investigate how cooperation between multiple energy-harvesting mobile devices can be facilitated to execute tasks in EH-D2D networks, which is to minimize the long-term system cost subject to the energy constraints of mobile device batteries.

3. System model

In this section, we firstly present the EH-D2D system architecture in which energy-harvesting mobile devices cooperate to execute computation tasks. We then introduce the task-queueing model, energy queue model and network model, respectively. The key notations used throughout this paper are listed in Table 1.

Table 1. Key notations

Symbols	Semantics
n	The number of mobile devices
MD_i	The ith mobile device
N_i^L	The number of low-performance CPU cores for MD_i
N_i^H	The number of high-performance CPU cores for MD_i
$F_{i,L}^{max}$	The maximum computation capacity of a low-performance CPU core
$F_{i,H}^{max}$	The maximum computation capacity of a high-performance CPU core
P_i^{ex}	The execution power of MD_i
P_i^{tr}	The transmission power of MD_i
P_i^{rx}	The received power of MD_i
E_i^{max}	The battery capacity of MD_i
T_{slot}	The time slot duration

$Q_i(\tau)$	The execution queue of MD_i in time slot τ
λ_i	The task arrival rate of MD_i
$a_i(\tau)$	The number of tasks arriving at MD_i in time slot τ
$b_i(\tau)$	The number of tasks executed on MD_i in time slot τ
$\mu_{ij}(\tau)$	The number of tasks offloaded from MD_i to MD_j in time slot τ
$\eta_{ij}(\tau)$	The number of tasks offloaded from MD_i to MD_j in time slot τ
E_i^{max}	The battery capacity of MD_i
$E_i^A(\tau)$	The available energy of MD_i in time slot τ
BW_i^{UL}	The uplink channel bandwidths of MD_i
$D_i(\tau)$	The number of tasks dropped on MD_i in time slot τ
$P_i(\tau)$	The battery energy penalty of MD_i in time slot τ
$R_i(\tau)$	The immediate reward of MD_i in time slot τ

3.1 EH-D2D system architecture

Fig. 1 shows an EH-D2D network that consists of a power beacon and n mobile devices $MD = \{MD_1, \ldots, MD_i, \ldots, MD_n\}$. Each mobile device MD_i can be denoted by a multi-tuple $MD_i = (N_i^L, N_i^H, F_{i,L}^{max}, F_{i,H}^{max}, P_i^{ex}, P_i^{tr}, P_i^{rx}, E_i^{max})$. Their definitions are as follows. N_i^L and N_i^H represent the number of low-performance and high-performance CPU cores, respectively. $F_{i,L}^{max}$ and $F_{i,H}^{max}$ are referred to as the maximum computation capacity of a low-performance core and a high-performance CPU core. P_i^{ex}, P_i^{tr} and P_i^{rx} are the execution, transmission and received power of MD_i, respectively. Finally, E_i^{max} is the battery capacity for mobile device MD_i. The mobile device's battery can be recharged by a power beacon.

Each mobile device MD_i generates a series of independent tasks. Each task can be characterized by two-tuples $t = (W, D)$, in which W denotes the task workload (GHz · s), and D denotes the data size (in MB) per unit workload, respectively. In particular, each mobile device MD_i contains an execution queue used to store the tasks offloaded from other mobile devices and those tasks to be executed locally.

Due to the limited computing resources and battery capacity of the mobile device, the arrival tasks can be selectively offloaded to other mobile devices nearby with sufficient energy and rich resources. Here, how to achieve collaborative task execution under limited battery capacity constraints is a critical issue. This paper mainly studies this problem and proposes an optimal collaborative task-execution scheme to minimize the long-term system cost.

In EH-D2D networks, we adopt a discrete-time model and logically divide the time horizon into time slots of equal duration T_{slot}. Each time slot's duration is $T_{slot} = 1S$. We denote the set of the time slot index τ by $\mathcal{T} = \{0, 1, \ldots, \tau, \ldots\}$. Each mobile device can be recharged by a power beacon in each time slot τ. When the available energy of mobile devices is insufficient to execute their arrival tasks, these tasks can be offloaded to mobile devices with sufficient energy. In each time slot τ, each mobile device makes a coordinated decision, including task offloading, task receiving and task executing, under available energy constraints.

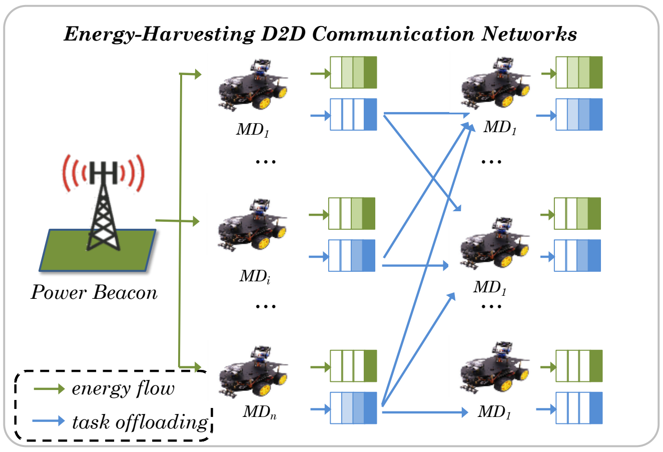


Fig. 1. Collaborative task execution in EH-D2D networks

3.2 Task-queueing model

In EH-D2D networks, each mobile device MD_i maintains an execution queue Q_i. The function of Q_i is to store the tasks waiting to be executed locally. These tasks in an execution queue Q_i are not offloaded again. These tasks mainly consist of offloaded tasks from other mobile devices and those generated by mobile device MD_i and assigned to execute locally. As in existing studies, the tasks' arrival process is assumed to follow a Poisson distribution with a parameter $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_i, \dots, \lambda_n)$. At the beginning of time slot τ, $\boldsymbol{A}(\tau) = (a_1(\tau), \dots, a_i(\tau), \dots, a_n(\tau))$ computation tasks arrive at the n mobile devices. Some of them are offloaded to other mobile devices, and some of them are assigned to execute locally. Let $\mu_{ij}(\tau)$ denote the number of tasks offloaded from MD_i to MD_j in time slot τ. Let $\eta_{ij}(\tau)$ denote the maximum number of tasks that MD_i can receive from MD_j in time slot τ. Thus, according to the above descriptions, the execution queue Q_i evolves according to Eq. (1):

$$Q_i(\tau+1) = \max\left[Q_i(\tau) + a_i(\tau)W - b_i(\tau)W - \sum_{i \neq j} \mu_{ij}(\tau)W, 0\right] + \sum_{j \neq i} \mu_{ji}(\tau)\, W \tag{1}$$

$$\mu_{ji}(\tau) \leq \eta_{ij}(\tau) \tag{2}$$

$$\sum_{j \in n} \mu_{ij}(\tau) = a_i(\tau) \tag{3}$$

$$\sum_{j \neq i, j \in n} \mu_{ij}(\tau) \leq a_i(\tau) \tag{4}$$

$$\sum_{j \neq i, j \in n} \mu_{ij}(\tau) + b_i(\tau) \leq Q_i(\tau) + a_i(\tau) \tag{5}$$

where $b_i(\tau)$ denotes the number of tasks executed on MD_i in time slot τ, and $\mu_{ji}(\tau)$ denotes the number of tasks offloaded from MD_j to MD_i in time slot τ. Eq. (2) guarantees that the number of tasks offloaded from MD_j to MD_i cannot exceed the maximum number of tasks that MD_i can receive from MD_j in time slot τ. Eq. (3) denotes that the number of arrival tasks in time slot τ is the sum of the number of offloaded tasks and the number of locally executed tasks in time slot τ. Eq. (4) denotes that the total number $\sum_{j \neq i, j \in n} \mu_{ij}(\tau)$ of offloaded tasks must be less than or equal to the number $a_i(\tau)$ of arrival tasks in time slot τ. Eq. (5) denotes that the sum of the number of offloaded tasks and the number of locally executed tasks for MD_i must be less than or equal to the sum of the number of tasks in Q_i and the number $a_i(\tau)$ of its arrival tasks. Since the length of the queue Q_i is constant, partial arrival tasks may be dropped due to insufficient storage space of the queue Q_i.

3.3 Energy-queueing model

The mobile device's battery can be recharged by a power beacon. The harvesting energy changes dynamically in different time slots. The main reason for this is that the mobility of mobile devices makes the channel gains between the power beacon and mobile devices change dynamically. The harvesting energy of MD_i in time slot τ can be denoted by $E_i^H(\tau) = \mu P h_i(\tau) T_{slot}$, where $\mu \in [0,1]$ denotes the energy-harvesting efficiency. P denotes the transmission power of the power beacon, and $h_i(\tau)$ denotes the channel gain between the power beacon and mobile device MD_i in time slot τ. The harvested energy can be used for receiving tasks, executing tasks and offloading tasks. Additionally, high-power mobile devices can cooperate in executing tasks offloaded from low-power ones. The available energy of MD_i at the beginning of each time slot τ can be presented by $E_i^A(\tau)$, which evolves according to Eq. (6):

$$E_i^A(\tau+1) = \max[E_i^A(\tau) - E_i^{ex}(\tau) - E_i^{tr}(\tau) - E_i^{rc}(\tau), 0] + E_i^H(\tau) \tag{6}$$

$$E_i^{ex}(\tau) + E_i^{tr}(\tau) + E_i^{rc}(\tau) \le E_i^A(\tau) \tag{7}$$

$$E_i^A(\tau) + E_i^H(\tau) \le E_i^{max} \tag{8}$$

$$E_i^H(\tau) \ge 0 \tag{9}$$

$$E_i^{ex}(\tau) \ge 0 \tag{10}$$

$$E_i^{tr}(\tau) \ge 0 \tag{11}$$

$$E_i^{rc}(\tau) \ge 0 \tag{12}$$

where $E_i^{ex}(\tau)$, $E_i^{tr}(\tau)$ and $E_i^{rc}(\tau)$ denote the amount of battery energy consumed by executing, transmitting and receiving tasks of MD_i in time slot τ, respectively. To guarantee sustainable operation, the sum of energy used for receiving tasks, executing tasks and offloading tasks must not exceed the currently available energy of mobile devices. This constraint condition can be denoted by Eq. (7). Eq. (8) guarantees that the available energy of MD_i and the recharged energy of MD_i cannot exceed the maximum battery capacity of MD_i. Eqs. (9)–(12) are the constraint conditions of the recharged energy and the energy consumed in executing tasks, offloading tasks and receiving tasks for MD_i, respectively.

3.4 Network model

In EH-D2D networks, mobile devices can communicate directly without interaction with edge servers. On account of mobile devices' mobility, the transmission rate of the wireless channel changes dynamically across different time slots. Let $r_{ij}^{UL}(\tau)$ and $r_{ij}^{DL}(\tau)$ denote the uplink and downlink transmission rates between MD_i and MD_j in time slot τ, respectively. Then, $r_{ij}^{UL}(\tau)$ and $r_{ij}^{DL}(\tau)$ can be calculated by Eqs. (13) and (14):

$$r_{ij}^{UL}(\tau) = BW_i^{UL} \log_2(1 + P_i^{tr} G_{ij}^{UL}(\tau)/\sigma^2), \forall i \in \mathcal{N} \tag{13}$$

$$r_{ij}^{DL}(\tau) = BW_i^{DL} \log_2(1 + P_j^{tr} G_{ij}^{DL}(\tau)/\sigma^2), \forall i \in \mathcal{N} \tag{14}$$

respectively, where BW_i^{UL} and BW_i^{DL} are the uplink and downlink channel bandwidths of MD_i, respectively; P_i^{tr} denotes the transmission power of MD_i; σ^2 is the Gaussian noise power; and $G_{ij}^{UL}(\tau)$ and $G_{ij}^{DL}(\tau)$ denote the uplink and downlink channel gain between MD_i and MD_j, respectively. The channel gain is mainly related to the communication distance between mobile devices. Due to the same communication distance of the uplink and downlink between MD_i and MD_j, the channel gains $G_{ij}^{UL}(\tau)$ and $G_{ij}^{DL}(\tau)$ can be calculated by $G_{ij}^{UL}(\tau) = G_{ij}^{DL}(\tau) = k(d_0/d_{ij})^{\theta}$, where α is the path-loss parameter; θ is the path-loss exponent; d_0 is the reference distance; and d_{ij} is the distance between MD_i and MD_j.

4 Multi-user collaborative task execution problem formulation

In the multi-user collaborative tasks execution scenario illustrated in Fig. 2, the prior knowledge of the system (including the execution queue state, the channel gain between mobile devices, the available battery energy, channel gain between the power beacon and mobile devices and the number of arrival tasks) is unknown in advance. To deal with such an optimization problem, we formulate it as a POMDP.

Firstly, the state space and action space are described, respectively. We then design the reward function of the collaborative task-execution problem. Lastly, we formulate this problem.

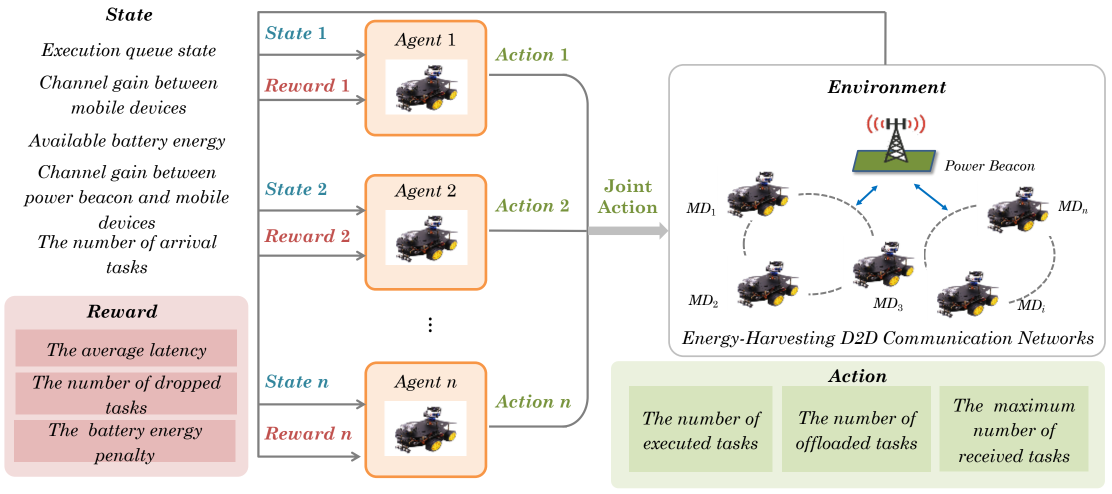

State
Execution queue state
Channel gain between mobile devices
Available battery energy
Channel gain between power beacon and mobile devices
The number of arrival tasks
Reward
The average latency
The number of dropped tasks
The battery energy penalty
State 1
Reward 1
Agent 1
Action 1
State 2
Reward 2
Agent 2
Action 2
State n
Reward n
Agent n
Action n
Joint Action
Environment
Power Beacon
MD1
MD2
MD3
MDi
MDn
Energy-Harvesting D2D Communication Networks
Action
The number of executed tasks
The number of offloaded tasks
The maximum number of received tasks

Fig. 2. Collaborative task execution based on the MADDPG

4.1 State space of collaborative task execution

Each mobile device MD_i acts as an agent to interact with the system environment [26][27]. At the beginning of time slot τ, each agent's observation space $O_i(\tau)$ (including the execution queue's state, the channel gain between MD_i and others, the available battery energy, the harvesting energy, and the number of arrival tasks) can be observed. This can be denoted by Eq. (15).

$$O_i(\tau) = [Q_i(\tau), G_i(\tau), E_i^A(\tau), E_i^H(\tau), a_i(\tau)] \tag{15}$$

where $Q_i(\tau)$ denotes the number of remaining tasks in the execution queue of MD_i in time slot τ; $G_i(\tau) = [G_{i1}(\tau), \dots, G_{i(i-1)}(\tau), G_{i(i+1)}(\tau), \dots, G_{in}(\tau)]$ denotes the channel gain between MD_i and other mobile devices except itself in time slot τ. $E_i^A(\tau)$ denotes the available energy of MD_i in the time slot. $E_i^H(\tau)$ denotes the amount of harvesting energy of MD_i in time slot τ, and $a_i(\tau)$ denotes the number of tasks arriving on MD_i in time slot τ. All agents' observation spaces constitute the whole system state. Thus, the system state $S(\tau) \in \mathbb{S}$ can be denoted by Eq. (16).

$$S(\tau) = [O_1(\tau), \dots, O_i(\tau), \dots, O_n(\tau)] \tag{16}$$

The system state $S(\tau)$ can be dynamically evolved according to transfer equations (1) and (6).

4.2 Action space of collaborative task execution

Each agent can choose an action $A_i(\tau)$ according to its own observation state $O_i(\tau)$. The actions for all agents form a joint action $\boldsymbol{A}(\tau)$. The action $A_i(\tau)$ consists of the number of locally executed tasks, the number of offloaded tasks and the maximum number of received tasks in time slot τ. It can be represented by Eq. (17)

$$\boldsymbol{A}(\tau) = [\boldsymbol{A}_1(\tau), \dots, \boldsymbol{A}_i(\tau), \dots, \boldsymbol{A}_N(\tau)]^T \tag{17}$$

$$\boldsymbol{A}_i(\tau) = [b_i(\tau), \boldsymbol{\mu}_i(\tau), \boldsymbol{\eta}_i(\tau)] \tag{18}$$

$$\boldsymbol{\mu}_i(\tau) = [\mu_{i1}(\tau), \dots, \mu_{i(i-1)}(\tau), \mu_{i(i+1)}(\tau), \dots, \mu_{in}(\tau)] \tag{19}$$

$$\boldsymbol{\eta}_i(\tau) = [\eta_{i1}(\tau), \dots, \eta_{i(i-1)}(\tau), \eta_{i(i+1)}(\tau), \dots, \eta_{in}(\tau)] \tag{20}$$

where $b_i(\tau)$ denotes the number of tasks executed on MD_i in time slot τ; $\boldsymbol{\mu}_i(\tau)$ denotes the vector for the number of tasks offloaded from MD_i to other $(n-1)$ mobile devices; and $\boldsymbol{\eta}_i(\tau)$ denotes the vector for the maximum number of tasks that MD_i can receive from other $(n-1)$ mobile devices.

Note that the total battery energy consumed by executing tasks, offloading tasks and receiving tasks in time slot τ cannot exceed the available energy of MD_i. The execution energy, offloading energy and receiving energy can be calculated as follows:

1) The execution energy consumption: Since the mobile device adopts an advanced dynamic voltage and frequency scaling (DVFS) technique, the CPU frequency can be automatically regulated in each time slot τ. Therefore, the computation capacity $C_i^{ex}(\tau)$ and computing power $P_i^{ex}(\tau)$ of MD_i in time slot τ can be denoted by Eq. (21) and Eq. (22):

$$C_i^{ex}(\tau) = N_i^L F_{i,L}(\tau) + N_i^H F_{i,H}(\tau) \tag{21}$$

$$P_i^{ex}(\tau) = \alpha_i (N_i^L F_{i,L}^3(\tau) + N_i^H F_{i,H}^3(\tau)) \tag{22}$$

where α_i is a constant related to the chip architecture. $F_{i,L}(\tau) \in [0, F_{i,L}^{max}]$ and $F_{i,H}(\tau) \in [0, F_{i,H}^{max}]$ are the realistic computation frequency of a low-performance and a high-performance CPU core. When mobile device MD_i decides to execute $b_i(\tau)$ tasks in time slot τ locally, we calculate the local execution energy consumption $E_i^{ex}(\tau)$ using Eq. (23):

$$E_i^{ex}(\tau) = b_i(\tau) P_i^{ex}(\tau) / C_i^{ex}(\tau) \tag{23}$$

2) The offloading energy consumption: When mobile device MD_i decides to offload $\mu_{ij}(\tau)$ number of tasks to other mobile devices in time slot τ, the number $\mu'_{ij}(\tau)$ of tasks actually offloaded can be denoted by Eq. (24). Thereby, the offloading energy consumption $E_i^{tr}(\tau)$ can be calculated using Eq. (25):

$$\mu'_{ij}(\tau) = \min\left(\min\left(r_{ij}^{UL}(\tau)/D, \eta_{ji}(\tau)\right), \mu_{ij}(\tau)\right) \tag{24}$$

$$E_i^{tr}(\tau) = P_i^{tr} \sum_{j\in n, j\neq i} \mu'_{ij}(\tau)\; D / r_{ij}^{UL}(\tau) \tag{25}$$

3) The receiving energy consumption: When the maximum number of tasks that MD_i can receive from MD_j in time slot τ is $\eta_{ij}(\tau)$, the number $\eta'_{ij}(\tau)$ of tasks actually received can be denoted by Eq. (26). Thereby, the receiving energy consumption $E_i^{rc}(\tau)$ can be calculated using Eq. (27)

$$\eta'_{ij}(\tau) = \min\left(\min\left(r_{ij}^{DL}(\tau)/D, \eta_{ij}(\tau)\right), \mu_{ji}(\tau)\right) \tag{26}$$

$$E_i^{rc}(\tau) = P_i^{rc} \sum_{j\in n, j\neq i} \eta'_{ij}(\tau)\; D / r_{ij}^{DL}(\tau) \tag{27}$$

Based on the above description, when the available energy of mobile device MD_i is $E_i^A(\tau)$, the sum of energy consumed by executing tasks, offloading tasks and receiving tasks must meet the following constraint:

$$E_i^{ex}(\tau) + E_i^{tr}(\tau) + E_i^{rc}(\tau) \le E_i^A(\tau) \tag{28}$$

4.3 Reward function

In a multi-agent collaborative scenario, each agent MD_i can have its own reward R_i. Each agent's reward R_i is conditioned only on its own observation $O_i(\tau)$ and action $A_i(\tau)$. The reward function R_i is defined to be the inverse of the system cost function C_i. We define C_i as the weighted sum of the average latency $Q_i(\tau)$ of task processing, the number $D_i(\tau)$ of dropped tasks and the battery energy penalty $P_i(\tau)$ in time slot τ, and $C_i(\tau)$ is denoted by Eq. (29). In order to encourage multiple agents to cooperate with one another, all agents use the whole system's reward R, which is the sum of each agent's reward R_i. We denote the reward R by Eq. (30).

$$C_i(\tau) = \omega_1 Q_i(\tau) + \omega_2 D_i(\tau) + \omega_3 P_i(\tau) \tag{29}$$

$$R(\tau) = \sum_{i=1}^{n} -C_i(\tau) \tag{30}$$

where ω_1, ω_2 and ω_3 are the weighted coefficients of $Q_i(\tau)$, $D_i(\tau)$ and $P_i(\tau)$, respectively. We can calculate the number of dropped tasks $D_i(\tau)$ according to Eq. (31).

$$D_i(\tau) = \max(\sum_{j=1}^{n} u_{ji}(\tau) - (|Q_i| + b_i(\tau) - Q_i(\tau)),0) \tag{31}$$

where $|Q_i|$ denotes the execution queue Q_i's length. $(|Q_i| + b_i(\tau) - Q_i(\tau))$ is the available space of the execution queue Q_i. In order to avoid the interruption of mobile applications due to exhaustion of mobile device's battery energy, we set the battery energy penalty threshold h_i. When the ratio of the mobile device's available energy $E_i^A(\tau)$ to the maximum battery capacity E_i^{max} is lower than h_i, the mobile device will be punished. The battery energy penalty $P_i(\tau)$ in time slot τ can be denoted by Eq. (32).

$$P_i(\tau) = \begin{cases} E_i^{ex}(\tau) + E_i^{tr}(\tau) + E_i^{rc}(\tau), & E_i^A(\tau) < h_i E_i^{max} \\ 0, & E_i^A(\tau) \geq h_i E_i^{max} \end{cases} \tag{32}$$

4.4 Problem formulation

The multi-user collaborative task-execution problem is formulated to be a POMDP. Its main goal is to maximize the whole system's long-term reward by dynamically coordinating multiple mobile devices to execute tasks. This problem is formulated as follows:

$$\text{Maximize:} \quad -R(\tau) \tag{33}$$

$$\text{Subject to:} \; \mu_{ji}(\tau) \leq \eta_{ij}(\tau) \tag{33a}$$

$$\sum_{j \in n} \mu_{ij}(\tau) = a_i(\tau) \tag{33b}$$

$$\sum_{j \neq i, j \in n} \mu_{ij}(\tau) \leq a_i(\tau) \tag{33c}$$

$$\sum_{j \neq i, j \in n} \mu_{ij}(\tau) + b_i(\tau) \leq Q_i(\tau) + a_i(\tau) \tag{33d}$$

$$b_i(\tau) \leq F_{max}/W \tag{33e}$$

$$u_{ij}(\tau) \leq r_{ij}^{UL}(\tau)/D \tag{33f}$$

$$u_{ji}(\tau) \leq r_{ij}^{DL}(\tau)/D \tag{33g}$$

$$b_i(\tau)W/N_i F_i(\tau) \leq T_{slot} \tag{33h}$$

$$\sum_{j \neq i, j \in n} \mu_{ij}(\tau)\, D/r_{ij}^{UL}(\tau) \leq T_{slot} \tag{33i}$$

$$\sum_{j \neq i, j \in n} \eta_{ij}(\tau)\, D/r_{ij}^{DL}(\tau) \leq T_{slot} \tag{33j}$$

$$E_i^{ex}(\tau) + E_i^{tr}(\tau) + E_i^{rc}(\tau) \leq E_i^A(\tau) \tag{33k}$$

$$E_i^{e}(\tau) \leq E_i^A(\tau) \tag{33l}$$

$$E_i^{tr}(\tau) \leq E_i^A(\tau) \tag{33m}$$

$$E_i^{rc}(\tau) \leq E_i^A(\tau) \tag{33n}$$

where Eq. (33) is the goal function of this paper. Eq. (33a) denotes that the number of tasks offloaded from MD_j to MD_i cannot exceed the maximum number of tasks MD_i receiving from MD_j. The constraint on the relationship between the number of arrival tasks and the number of offloaded tasks can be denoted by Eqs. (33b) and (33c). The constraint on the sum of the number of offloaded tasks and the number of executed tasks can be denoted by Eq. (33d). All equations between Eqs. (33e) and (33h) denote the constraints of the network transmission rate on the number of executed tasks, the number of offloaded tasks and the maximum number of received tasks, respectively. Eqs. (33i) and (33j) denote the constraints of time slot duration on the number of executed tasks, the number of offloaded tasks, and the maximum number of received tasks. Eq. (33k) guarantees that the sum of the execution energy, the offloading energy and the receiving energy cannot exceed the available battery energy of the mobile device. Eqs. (33l)–(33n) denote the constraints on the execution energy, offloading energy, and receiving energy, respectively.

5. Algorithm implementation

To solve the multi-user collaborative task-execution problem, we design the CACTE scheme based on the MADDPG algorithm [28][29][30] to maximize the whole system's long-term reward. The multi-agent policy gradient algorithm is a centralized training and decentralized execution framework in which each agent can learn the cooperation policy and improve the efficiency of the system. The CACTE scheme based on the MADDPG algorithm is described in detail.

The MADDPG algorithm, based on a policy gradient, consists of multiple collaborative agents. Each agent MD_i consists of the evaluated networks, the target networks and the replay memory Ω_i. Its evaluated networks mainly include an evaluated actor-network $\mu_i(O_i(\tau)|\theta_i{}^{\mu_i})$ and an evaluated critic network $Q_i(O_i(\tau), A_i(\tau)|\theta_i{}^{Q_i})$. Based on the local observation $O_i(\tau)$, the evaluated actor-network generates an action $A_i(\tau)$. Based on the local observation $O_i(\tau)$ and the action $A_i(\tau)$, the evaluated critic network calculates the evaluated Q_i' value. The target actor-network and target critic network are a copy of their own evaluated networks, $\mu_i'(O_i(\tau)|\theta_i{}^{\mu_i'})$ and $Q_i'(O_i(\tau), A_i(\tau)|\theta_i{}^{Q_i'}))$, respectively. Based on the target networks, the target Q_i value can be calculated. Each agent MD_i's transition experience $(O_i(\tau), A_i(\tau), R_i(\tau), O_{-i}(\tau))$ is stored in its own replay memory Ω_i. Based on its transition experiences, the learned networks and the target networks of each agent MD_i are updated. The proposed multi-agent reinforcement learning-based approach mainly consists of the interaction stage, training stage and testing stage. Its detailed processes are described in Algorithm 1.

During the interaction stage, we firstly initialize each agent's learned networks, target networks and memory capacity (lines 1–3). In addition, we initialize each agent's environment parameters, such as the execution queue, the channel gain between mobile devices, the available energy, and the channel gain between the power beacon and the mobile devices (line 4). We then initialize each agent's local observation $O_i(\tau)$ (line 6). Based on the initial state $O_i(\tau)$, each agent determines its action $A_i(\tau)$ (line 11). After taking the action $A_i(\tau)$ at the current local observation $O_i(\tau)$, each agent i obtains the immediate reward $R_i(\tau)$ and observes its new local state $O_{-i}(\tau+1)$ in the next time slot $(\tau+1)$. Lastly, each agent i stores its transition experience $(O_i(\tau), A_i(\tau), R_i(\tau), O_{-i}(\tau+1))$ into its replay memory Ω_i.

During the entire training stage, each agent i firstly randomly samples a mini-batch of transition experiences $\widetilde{U}_i$ from its replay memory Ω_i (line 15). Each agent i then computes its target action value $\mu_i'\left(O_{-i}(\tau+1)|\theta_i{}^{\mu_i'}\right)$ and target Q value $Q_i'\left(O_{-i}(\tau+1), \mu_i'\left(O_{-i}(\tau+1)|\theta_i{}^{\mu_i'}\right)\middle|\theta_i{}^{Q_i'}\right)$ for its next observation state $O_{-i}(\tau+1)$, respectively. The target Q value for its observation $O_i(\tau)$ is updated to be $y_i(\tau) = R_i(\tau) + \gamma Q_i'\left(O_{-i}(\tau+1), \mu_i'\left(O_{-i}(\tau+1)|\theta_i{}^{\mu_i'}\right)\middle|\theta_i{}^{Q_i'}\right)$ in its target critic network. The value $y_i(\tau)$ is transmitted to its evaluated critic network. After receiving y_i, the evaluated critic network is updated by minimizing the loss function $L_i = \sum_{\widetilde{U}_i}(y_i(\tau) - Q_i(O_i(\tau), A_i(\tau)|\theta_i{}^{Q_i}))^2/\widetilde{U}_i$. Based on the parameter $\theta_i{}^{Q_i}$, the actor-network parameter $\theta_i{}^{\mu_i}$ is optimized with the policy gradient $\nabla_{\theta^{\mu}}\mathcal{J}$. Lastly, we adopt the *soft update* scheme to update the target network parameters to achieve learning stability.

During the testing stage, the learned network parameters are loaded first. Each agent's local state is observed and fed into its evaluated actor-network. Each agent's action is then generated based on its trained actor-network. According to each agent's state and action, its immediate reward can be calculated. In order to make multiple agents cooperate with one another, the immediate reward for the whole system is obtained by summing all agents' rewards.

Algorithm 1: CACTE Scheme

```
BEGIN
01: For each agent $i$, initialize the weight $\theta_i^{Q_i}$ of the evaluate critic network $Q_i(O_i, A_i|\theta_i^{Q_i})$ and
    the weight $\theta_i^{\mu_i}$ of the evaluate actor-network $\mu_i(O_i|\theta_i^{\mu_i})$;
02: For each agent $i$, initialize the weight $\theta_i^{Q_i'}$ of the target critic network $Q_i'(O_{-i}, A_{-i}|\theta_i^{Q_i'})$ with
    $\theta_i^{Q_i'} = \theta_i^{Q_i}$ and the weight $\theta_i^{\mu_i'}$ of the target actor-network $\mu_i'(s|\theta_i^{\mu_i'})$ with $\theta_i^{\mu_i'} = \theta_i^{\mu_i}$;
03: For each agent $i$, initialize its experience replay memory $\Omega_i$ with size $U_i$ and the mini-batch
    $\tilde{\Omega}_i \subset \Omega_i$ with size $\tilde{U}_i$;
04: For each agent $i$, initialize the task arrival rate $\lambda_i$, the execution queue $Q_i$, the channel gain $g_{ij}$
    between mobile devices, the available energy $E_i$, and the channel gain between mobile devices
    and power beacon.
05: for cur_ep = 1, MAX_EPISODES do
06:   For each agent $i$, reset its local environment and observe its initial local observation $O_i(\tau)$.
07:   Observe the whole system's initial state $S(\tau) = [O_1(\tau), \dots, O_i(\tau), \dots, O_n(\tau)] \in \mathbb{S}$ which is a
      two-dimensional array consisting of $n$ agents' initial local state;
08:   Initial the total reward $ep\_reward = 0$;
09:   For each agent $i$, initialize its noise object $orn_i$ to explore its action;
10:   for $\tau = 1$, MAX_EP_STEPS do
11:     For each agent $i$, select action $A_i(\tau) = \mu_i(O_i(\tau)|\theta_i^{\mu_i}) + orn_i$ based on its current policy
        $\mu_i(O_i(\tau)|\theta_i^{\mu_i})$ and exploration noise $orn_i$; Observe other $(n-1)$ agents' joint action
        $act\_n_i(\tau) = [A_{-1}(\tau), \dots, A_{-(i-1)}(\tau), A_{-(i+1)}(\tau), \dots, A_{-n}(\tau)]$;
12:     For each agent $i$, execute its actions $A_i(\tau)$, calculate its immediate reward $R_i(\tau)$ and
        observe its new state $O_{-i}(\tau)$;
13:     For each agent $i$, store the experience transition $(O_i(\tau), A_i(\tau), R_i(\tau), O_{-i}(\tau+1))$ into its
        replay memory $\Omega_i$;
14:     Update the long-term system reward $ep\_reward \mathrel{+}= \sum_{i=1}^{n} R_i(\tau)$ and the current system
        state $S(\tau) = S\_(\tau+1)$;
15:     For each agent $i$, randomly choose a mini-batch of experiences $\tilde{U}_i$ from $\Omega_i$; Compute the
        actor value $A_{-i}(\tau+1)$ for next state $O_{-i}(\tau+1)$; Observe other $(n-1)$ agents' joint
        action $act\_n_i(\tau+1) = [A_{-1}(\tau+1), \dots, A_{-(i-1)}(\tau+1), A_{-(i+1)}(\tau+1), \dots, A_{-n}(\tau+1)]$;
16:     For each agent $i$, compute its target actor value $\mu_i'\left(O_{-i}(\tau+1)|\theta_i^{\mu_i'}\right)$ and the target $Q$
        value $Q_i'\left(S(\tau+1), act\_n_i(\tau+1), \mu_i'\left(O_{-i}(\tau+1)|\theta_i^{\mu_i'}\right)\middle|\theta_i^{Q_i'}\right)$ for the next state $s\_(\tau+1)$;
17:     For each agent $i$, update its target $Q$ value $y_i(\tau)$:
            $y_i(\tau) = R_i(\tau) + \gamma Q_i'\left(S(\tau+1), act\_n_i(\tau+1), \mu_i'\left(O_{-i}(\tau+1)|\theta_i^{\mu_i'}\right)\middle|\theta_i^{Q_i'}\right)$;
18:     For each agent $i$, update its target critic network by minimizing the loss $L$:
            $L = \frac{1}{\tilde{U}} \sum_{\tau} (y_i(\tau) - Q_i(S(\tau), act\_n_i(\tau), A_i(\tau)|\theta_i^{Q_i}))^2$;
19:     For each agent $i$, update its target actor-network:
            $\nabla_{\theta_i^{\mu_i}} \mathcal{J} \approx \frac{1}{\tilde{U}} \sum_{\tau} \nabla_{A_i(\tau)} Q_i\left(O_i(\tau), A_i(\tau)|\theta_i^{Q_i}\right)|_{O_i(\tau), \mu_i(O_i(\tau))} \nabla_{\theta_i^{\mu_i}} \mu_i(O_i(\tau)|\theta_i^{\mu_i})|_{O_i(\tau)}$
20:     Update the target network:
            $\theta_i^{Q_i'} \leftarrow \mathcal{T}_i \theta_i^{Q_i} + (1 - \mathcal{T}_i)\theta_i^{Q_i'}$
            $\theta_i^{\mu_i'} \leftarrow \mathcal{T}_i \theta_i^{\mu_i} + (1 - \mathcal{T}_i)\theta_i^{\mu_i'}$
```

```
21:     end for
22: end for
END
```

6. Performance evaluations

To evaluate the performance of the CACTE scheme, we conducted extensive simulation experiments. We first explain the related parameter settings in detail and then introduce the four baseline algorithms (including Local, Random, ECLB and CCLB). Finally, we discuss our comparison of the proposed CACTE scheme with these four baseline algorithms in terms of the task arrival rate, the battery capacity and so on, and we analyze the experimental results in detail.

6.1 Parameter settings

This paper focuses on the scenario where n mobile devices cooperate with one another to execute the computation tasks in EH-D2D networks. To solve the multi-user collaborative task-execution problem, we propose the CACTE scheme and implement it in Python with TensorFlow 2.0. We list the parameter configurations of different types of mobile devices in Table 2. Experimental parameters are assigned in consistence with numerous existing studies, such as [31][32] and [33].

Table 2. Parameter configurations of different types of mobile devices

$Type_i$	N_i^L	N_i^H	$F_{i,L}^{max}$ (GHz)	$F_{i,H}^{max}$ (GHz)
$Type_1$	4	4	1.8	2.6
$Type_2$	4	4	1.7	2.9
$Type_3$	6	2	1.7	2
$Type_4$	4	2	1.6	2.5

In our model, the number of mobile devices is initialized to be $n = 4$. The four mobile devices are of different types. Initially, the types of mobile devices MD_1, MD_2, MD_3 and MD_4 are set to be $Type_1$, $Type_2$, $Type_3$ and $Type_4$, respectively. This means that the number of low-performance cores of MD_1, MD_2, MD_3 and MD_4 are $N_1^L = 4$, $N_2^L = 4$, $N_3^L = 6$ and $N_4^L = 4$, respectively. The maximum CPU-cycle frequencies of four low-performance CPU cores are $F_{1,L}^{max} = 1.8$GHz, $F_{2,L}^{max} = 1.7$GHz, $F_{3,L}^{max} = 1.7$GHz and $F_{4,L}^{max} = 1.6$GHz, respectively. At the same time, the number of high-performance cores of MD_1, MD_2, MD_3 and MD_4 are $N_1^H = 4$, $N_2^H = 4$, $N_3^H = 2$ and $N_4^H = 2$, respectively. The maximum CPU-cycle frequencies of four high-performance CPU cores are $F_{1,H}^{max} = 2.6$GHz, $F_{2,H}^{max} = 2.9$GHz, $F_{3,H}^{max} = 2$GHz and $F_{4,H}^{max} = 2.5$GHz, respectively. The CPU frequency level of each mobile device can be dynamically adjusted by the DVFS technique. The constant α_i of each mobile device is set as $\alpha_i = 0.1125 \ \mathrm{W/(GHz)^3}$ [14][34].

As for the task model, we assume that the task arrival process for each mobile device will follow a Poisson distribution with λ_i. Each mobile device's task arrival rate λ_i varies within the range $[0,12]$. Initially, the task arrival rates for four mobile devices are set as $\lambda_1 = 3$, $\lambda_2 = 1$, $\lambda_3 = 7$ and $\lambda_4 = 10$, respectively. Each task's workload W varies within the range $[0.6, 1.4]$Gycles, with an initial value of $W = 1\mathrm{GHz \cdot s}$. Each task's data size is $D = 1\mathrm{MB}$.

In our network communication model, the communication distance between any two mobile devices is set to within 200 m. The uplink and downlink channel bandwidths of each mobile device are $BW_i^{UL} =$

10MHz and $BW_i^{DL} = 10\text{MHz}$, respectively. The additive white Gaussian noise power σ^2 is set to -174dbm/Hz. In addition, we set the path-loss constant k to 0.01, the path-loss exponent θ to 4, and the reference distance d_0 to 1m [31] [35], respectively. The transmission power and receiving the power of each mobile device are $P_i^{tr} = 0.25\text{W}$ and $P_i^{rc} = 0.1$ [36].

The weights for the average latency of task processing, the number of dropped tasks and the power penalty are $\omega_1 = 1$, $\omega_2 = 15$ and $\omega_3 = 90$.

In our deep neural network model, we construct the evaluated actor-network, the evaluated critic network, the target actor-network and the target critic network for each agent. The structures of the evaluated actor-network and the evaluated critic network are the same as those of their own target networks, respectively. The evaluated actor-network is composed of two hidden layers. There are 30 neurons in each hidden layer. The evaluated critic network is composed of three hidden layers. There are also 30 neurons in each hidden layer. We adopt the Adam optimizer to update the target networks, in which the actor-network's learning rate is 0.0001, and the critic network's learning rate is 0.003. The reward discount factor γ is set to be 0.9. Lastly, the capacity of the replay memory is set to be 10000, and the mini-batch size to be 16.

6.2 Experimental results

To validate the performance of CACTE, we implement four baseline algorithms (including Local, Random, ECLB and CCLBs) and compare these with CACTE in terms of performance metrics, such as the total cost, the total average delay in task processing and the total number of dropped tasks.

- **Local Execution (Local):** The arrival tasks for each mobile device are executed locally. There is no cooperation between different mobile devices.
- **Random Cooperation (Random):** In each time slot τ, each mobile device randomly selects another one to execute its tasks collaboratively.
- **Energy Capacity Load Balance (ECLB):** In each time slot τ, all mobile devices' arrival tasks are distributed evenly according to their battery energy. The number of tasks offloaded from MD_i to MD_j can be denoted by Eq. (33).

$$off_{ij}(\tau) = \left\lfloor \frac{E_j^A(\tau)}{\sum_{k=0}^{n} E_k^A(\tau)} \right\rfloor a_i(\tau) \tag{33}$$

  Based on the above description, the residual tasks $a_i(\tau) - \sum_{j\in n, j\neq i} off_{ij}(\tau)$ can be locally executed on the ith mobile device.
- **Computation Capacity Load Balance (CCLB):** In each time slot τ, the arrived tasks are evenly distributed to all mobile devices according to their computing capacity. The number of tasks distributed to the jth mobile device by the ith mobile device can be denoted by Eq. (34)

$$off_{ij}(\tau) = \left\lfloor \frac{N_i F_i^{max}}{\sum_{i=0}^{n} N_i F_i^{max}} \right\rfloor a_i(\tau) \tag{34}$$

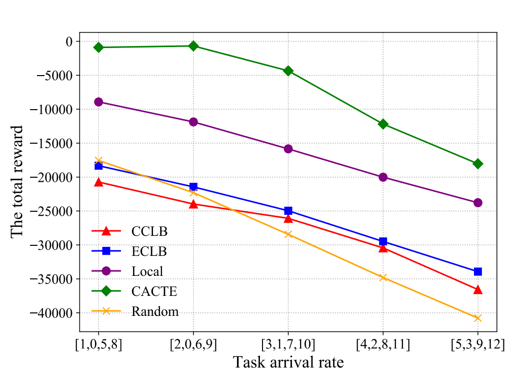
  Note that $off_{ij}(\tau)$ is an integer, and the number of tasks executed on MD_i is $a_i(\tau) - \sum_{j\in n, j\neq i} off_{ij}(\tau)$.

6.2.1 The impact of task arrival rate $\boldsymbol{\lambda}$

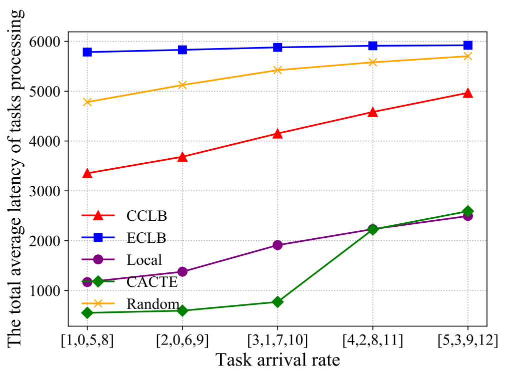
To examine the effect of $\boldsymbol{\lambda}$, we vary $\boldsymbol{\lambda}$ from [1, 0, 5, 8], [2, 0, 6, 9], [3, 1, 7, 10], [4, 2, 8, 11] to [5, 3, 9, 12] with an increment of [1, 1, 1, 1]. In Fig. 3(a), we can see that the total reward obtained by the CACTE scheme outperforms the Local algorithm, Random algorithm, ECLB algorithm and CCLB algorithm. This is because, in the CACTE scheme, each mobile device can make an optimal coordinated decision, according to the changing environmental factors (e.g., the number of arrival tasks, available space in the execution queue, transmission rate, available battery energy and charging energy). Hence, each mobile device can process many more tasks and obtain a higher long-term reward. In contrast, the Local, Random, ECLB and CCLB algorithms cannot perceive the dynamically changing environment in order to make an optimal decision for collaborative task execution. In addition, it is evident that when the task arrival rate increases, the total rewards of the CACTE scheme and four baseline algorithms all decrease gradually. This is because the system's total processing capacity is bounded by mobile devices' limited computing, network and battery energy resources. Therefore, the higher the task arrival rate, the more tasks arrive, and the more dropped tasks result, accordingly.

In Figs. 3(b) and (c), we see that when the task arrival rate increases, the total average latency and the total number of dropped tasks for the CACTE scheme and four baseline algorithms increase. Moreover, the CACTE scheme can achieve lower total average latency and fewer dropped tasks than the other four baseline algorithms. This is due to the CACTE scheme being able to take advantage of mobile devices' limited computing, network and battery energy resources to execute more tasks collaboratively.

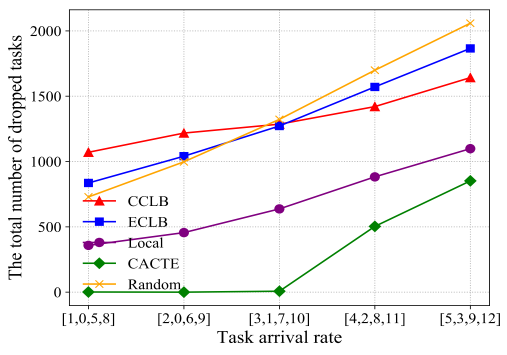
In Fig. 3(d), we see that the total battery energy penalty of the CACTE scheme is lower than that of the Local, Random and CCLB algorithms, but it is higher than that of the ECLB. This is because the ECLB can evenly distribute all arrival tasks according to the available battery energy. The available battery energy of each mobile device is higher than the battery energy penalty threshold, and therefore the total battery energy penalty is the lowest, with a value of 0. In general, this result indicates that in comparison with the other four algorithms, the CACTE scheme can achieve the lowest total average delay, the lowest total number of dropped tasks and a lower total battery energy penalty.

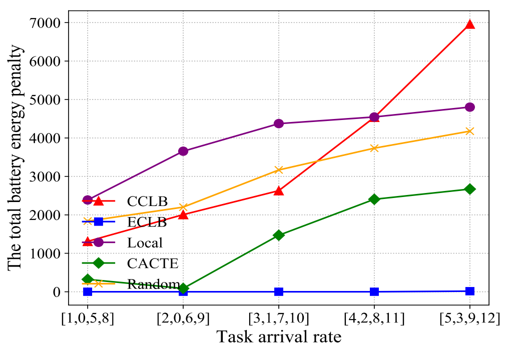


(a) The total reward



(b) The total average latency of task processing





(c) The total number of dropped tasks (d) The total battery energy penalty

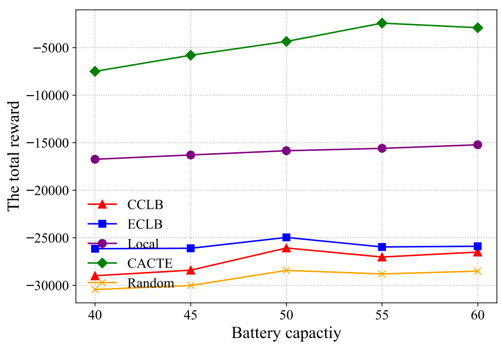
Fig. 3. The impact of task arrival rate

6.2.2 The impact of mobile devices' battery capacity $E_i^A(\tau)$

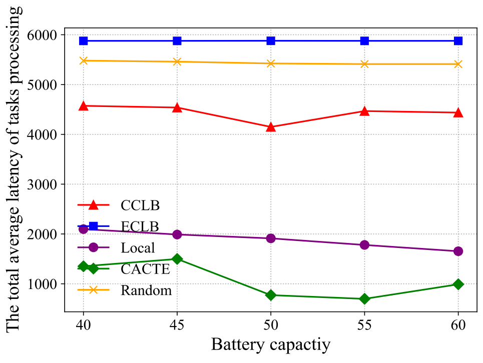
To investigate the performance impact of the mobile devices' battery capacity $E_i^A(\tau)$, we vary mobile devices' battery capacity $E_i^A(\tau)$ from 40 energy units to 60 energy units, with an increment of 5. As shown in Fig. 4(a), we can see that when the battery capacity is less than 55, with an increase in battery capacity, the total rewards of the CACTE scheme and four baseline algorithms all increase. This is because when a mobile device's battery capacity increases, the computing capacity of the mobile device improves as well. High-performance mobile devices can process more tasks and drop fewer tasks, thereby obtaining a higher total reward. However, the curves of all algorithms are flat when the mobile device's battery capacity is equal to or larger than 55. This is because when the mobile device's battery capacity reaches 55, the battery energy is no longer a bottleneck restricting collaborative task execution. With such computing and network capacity, the system can process the maximum number of tasks. Additionally, in Fig. 4(a), we can confirm that the CACTE scheme achieves a higher total reward than the other four baseline algorithms because the CACTE scheme can take advantage of mobile devices' limited resources to execute many more tasks collaboratively.

Fig. 4(b) exhibits the total average latency of task processing for all algorithms. In Fig. 4(b), we can see that when the battery capacity of the mobile device increases, the total average latency of task processing gradually decreases and becomes stable. This is because when the battery capacity is less than or equal to 55, the tasks cannot be executed collaboratively on time due to insufficient battery energy. With an increase in battery capacity, collaborative task execution is no longer restricted due to insufficient battery energy. With such battery capacity, the system can reach its upper processing limit. Meanwhile, the total average latency of tasks processing for the CACTE scheme is lower than that of the other four algorithms. The reason for this is the same as that given above.

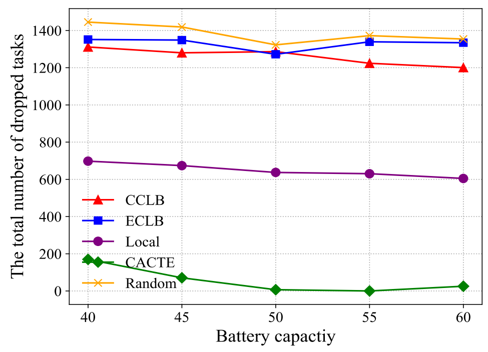
In Fig. 4(c), we can observe that when the battery capacity of the mobile device increases, the CACTE scheme and the other four baseline algorithms drop fewer tasks. Specifically, when the battery capacity is equal to or greater than 50, there are almost no dropped tasks under the CACTE scheme. Moreover, the CACTE scheme has fewer dropped tasks than the other four baseline algorithms. An explanation for this is that under the CACTE scheme, each mobile device makes an optimal coordinated decision according to its observed system state, thereby enabling multiple mobile devices with limited resources to cooperate with one another to execute more tasks. In Fig. 4(d), the total battery energy is lower than that of the Local, Random and CCLB algorithms, but it is higher than that of the ECLB. The reason for this is the same as that discussed in Section 6.3.1.



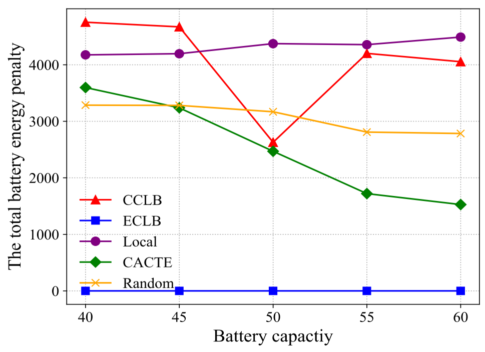
(a) The total reward



(b) The total average latency of task processing



(c) The total number of dropped tasks



(d) The total battery energy penalty

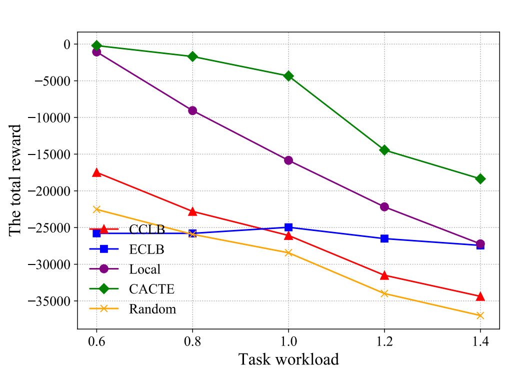
Fig. 4. The impact of mobile devices' battery capacity

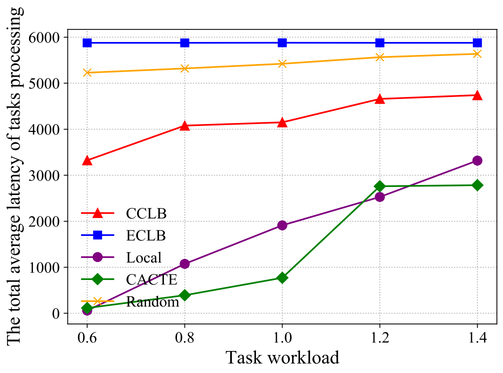
6.2.3 The impact of task workload W

To examine the effect of W, we vary W in the range of 0.6 to 1.4. The experimental results are given in Fig. 5. It is evident that when task workload W increases, the total reward obtained by the CACTE scheme gradually decreases (as in Fig. 5(a)). Clearly, with limited computing, network and battery energy resources, when the task workload increases, the system will process fewer tasks and drop more tasks in a time slot, thereby achieving a lower total reward. Additionally, it can be seen that the CACTE scheme achieves a higher total reward than the other four baseline algorithms. This is because that the CACTE scheme takes full advantage of mobile devices' computing, network and battery energy resources to execute more tasks collaboratively.

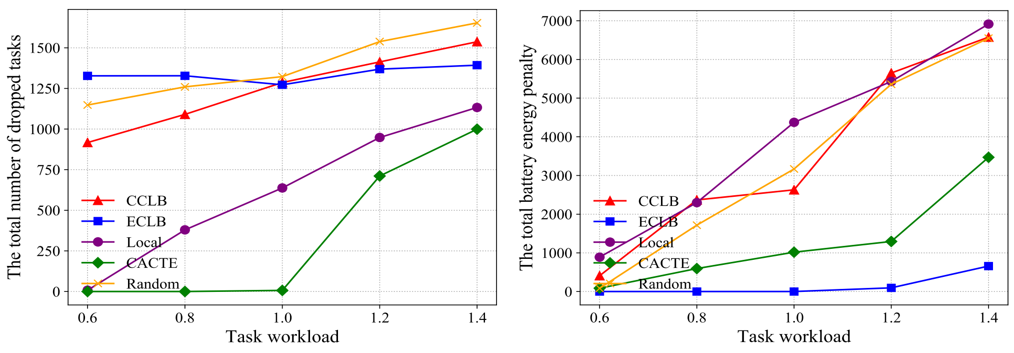
In Fig. 5(b), we can see that the total average latency of the CACTE scheme firstly increases and then becomes stable when the task workload increases. When the task workload increases, the system processes fewer tasks in a time slot, thereby resulting in a higher total average latency. Moreover, we also see that the total average latency of the CACTE scheme is lower than that of the other four baseline algorithms. This is due to the CACTE scheme being able to perform an optimal collaborative execution policy according to the environmental dynamics, thereby processing many more tasks.

In Fig. 5(c), we can observe that when the task workload increases, the system drops more tasks. When the size of the mobile device's execution queue is constant, the system processes fewer tasks in a time slot, thereby dropping more tasks. Moreover, in comparison with the other four algorithms, fewer tasks are dropped by the CACTE scheme. This is because that more tasks are collaboratively executed by multiple energy-harvesting mobile devices. In Fig. 5(d), it can be seen that the total battery energy of the CACTE scheme is higher than that of the ECLB but is lower than that of the Local, Random and CCLB algorithms. The reason for this is the same as that given in Section 6.3.1.





(a) The total reward (b) The total average latency of task processing



(c) The total number of dropped tasks (d) The total battery energy penalty

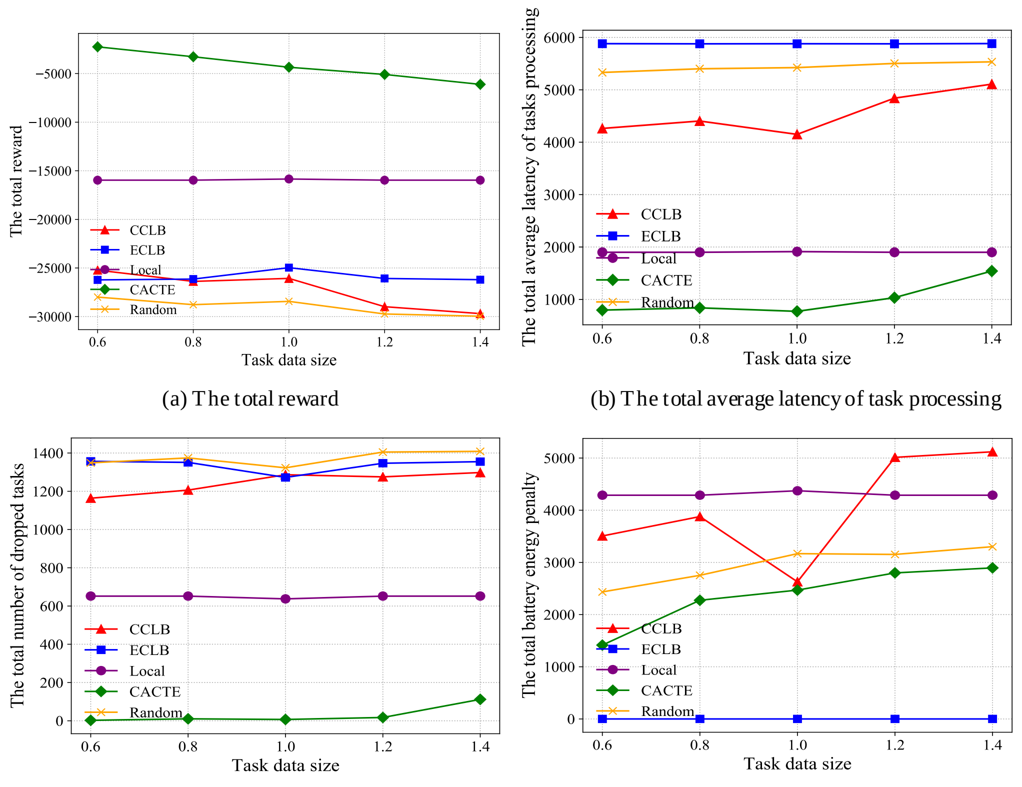
Fig. 5. The impact of task workload

6.2.4 The impact of task data size D

To examine the effect of the parameter D, we vary D in the range of 0.6 to 1.4. The experimental results are given in Fig. 6. In Fig. 6(a), it can be seen that the total reward obtained by the CACTE scheme gradually decreases with an increase in task data size. The main reason for this is that when the task data size increases, the system with limited network and battery energy resources transmits fewer computation tasks in a time slot, which results in a larger average latency of task processing. Specifically, the total reward of the CACTE scheme is higher than that of the four baseline algorithms. This is due to the CACTE scheme being able to make an optimal decision to collaboratively execute many more computation tasks according to dynamic changes in the system. Therefore, the long-term reward obtained by the CACTE scheme is higher than that obtained by the other four algorithms.

In Fig. 6(b), it can be seen that when the task data size increases, the total average latency of task processing increases. The main reason for this is that when the data size of the computation task increases, the number of transmitted tasks decreases, which results in fewer tasks being executed collaboratively. Moreover, the total average latency of task processing for the CACTE scheme is lower than that for the other four algorithms. This is because the CACTE scheme can take full advantage of limited resources to execute more tasks collaboratively.

In Fig. 6(c), it can be seen that when the task data size increases, the total number of dropped tasks increases. This is because when fewer tasks are executed collaboratively, more tasks are dropped. In particular, the CACTE scheme drops fewer tasks in comparison with the other four baseline algorithms because more tasks are executed collaboratively under the CACTE scheme compared with the other four algorithms. In Fig. 6(d), the total battery energy is lower than that of the Local, Random and CCLB algorithms, but it is higher than that of the ECLB. The reason for this is the same as that discussed above.



(a) The total reward (b) The total average latency of task processing

(c) The total number of dropped tasks (d) The total battery energy penalty

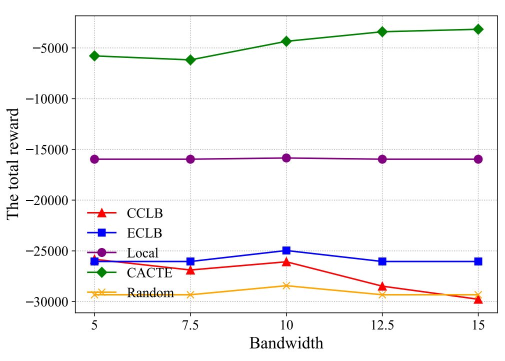
Fig. 6. The impact of task data size

6.2.5 The impact of bandwidth $BW^{UL}(\tau)$

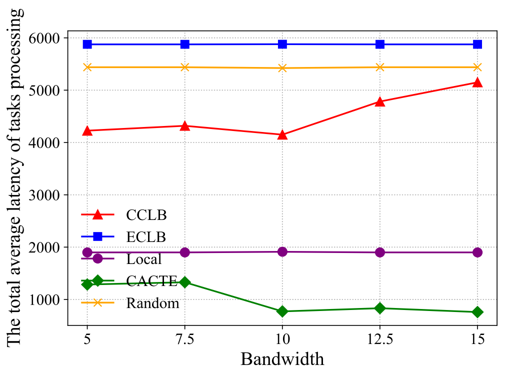
In our experiments, the bandwidth of the mobile devices' uplink is set to be equal to their downlink. Fig. 7 shows the effect of bandwidth $BW^{UL}(\tau)$ on the total reward. The experimental results are plotted when the bandwidth $BW^{UL}(\tau)$ varies from 5 MHz to 15 MHz, with an increase of 0.25. Fig. 7(a) shows that when the bandwidth increases, the total reward of the CACTE scheme increases and stabilizes. This is because that when the bandwidth of a mobile device is less than or equal to 10, the larger the bandwidth is, the higher the transmission rate between mobile devices is. Hence, more tasks can be executed collaboratively to obtain a higher total reward. However, when $BW^{UL}(\tau)$ is larger than 10, the curves of the CACTE scheme are flat. An explanation for this is that the transmission rate between mobile devices is no longer a bottleneck restricting collaborative task execution when the bandwidth is larger than 10. With the current computing and network capacity, the system reaches its upper processing limit and can process the maximum number of tasks. Moreover, the CACTE scheme achieves a higher total reward in comparison with the other four baseline algorithms. This is due to the CACTE scheme taking advantage of system resources to execute more tasks collaboratively. Therefore, the long-term reward obtained by the CACTE scheme is higher than that obtained by the other four algorithms.

In Fig. 7(b), we can see that when the bandwidth increases, the total average latency of task processing gradually decreases and stabilizes. An explanation for this is that when the bandwidth increases, the system transmits mores tasks in a time slot, thereby executing more tasks collaboratively and resulting in the lower average latency of task processing. Moreover, the total average latency of task processing for the CACTE scheme is lower than that for the other four algorithms. This is because the CACTE scheme can use limited resources to execute more tasks collaboratively in a time slot.

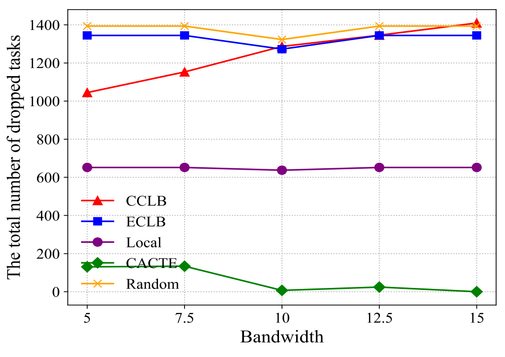
In Fig. 7(c), we can see that when the bandwidth increases, the total number of dropped tasks decreases. This is because when the bandwidth increases, the system executes more tasks collaboratively, thereby dropping fewer tasks. The CACTE scheme drops fewer tasks than the other four baseline algorithms because it executes more tasks collaboratively. In Fig. 7(d), we can see that the CACTE scheme obtains lower total battery energy than the Local, Random and CCLB algorithms, and it obtains higher total battery energy than the ECLB algorithm. The reason for this is explained above.



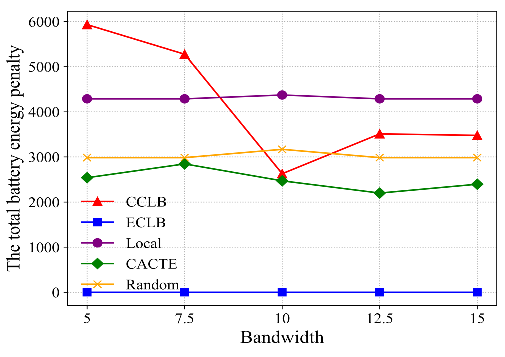
(a) The total reward



(b) The total average latency of task processing



(c) The total number of dropped tasks



(d) The total battery energy penalty

Fig. 7. The impact of bandwidth

6.2.6 The impact of the number n of mobile devices

In order to examine the effect of n, experiments with n being 3, 4, 5 and 6 are conducted, respectively. The related experimental results are shown in Tables 3 to 6. In the collaborative task-execution scenario consisting of three mobile devices with $Type_1$, $Type_2$ and $Type_4$ parameter configurations, the CACTE scheme achieves a higher total reward, the lower total average latency of tasks processing and the lower total number of dropped tasks, compared with the Local, Random, CCLB and ECLB algorithms. This is due to the CACTE scheme taking advantage of the limited resources to execute more tasks collaboratively. When more tasks are executed, fewer tasks are dropped, resulting in a higher long-term reward. Meanwhile, in the collaborative task-execution scenario consisting of four mobile devices with $Type_1$, $Type_2$, $Type_3$ and $Type_4$ parameter configurations, we can see that the CACTE scheme also outperforms the other four algorithms. The experimental results indicate that in the case of a different number of mobile devices, the CACTE scheme can achieve the highest long-term reward, the minimum average latency of task processing and the fewest number of dropped tasks, compared with the Local, Random, CCLB and ECLB algorithms. Moreover, in the scenario, which consists of two mobile devices with $Type_4$ parameter configurations, and the other three mobile devices with $Type_1$, $Type_2$ and

$Type_3$ parameter configurations, we can see the same phenomenon. In the scenario, which consists of two mobile devices with $Type_1$ parameter configuration, two mobile devices with $Type_4$ parameter configuration, and other two mobile devices with $Type_2$, $Type_3$ parameter configurations, the same phenomenon can be also observed. The reason for this is the same as that discussed above.

Table 3. The impact of three mobile devices

Performance Metrics / Different Algorithms	The Total Reward	The Total Average Latency	The Total Number of Dropped Tasks
Local	-12379.5	1339.1	565.8
Random	-18355.6	3856.1	812.9
CCLB	-15203.9	3415.8	608.2
ECLB	-16654.9	4302.4	823.5
CACTE	**-1505**	**623.3**	**0**

Table 4. The impact of four mobile devices

Performance Metrics / Different Algorithms	The Total Reward	The Total Average Latency	The Total Number of Dropped Tasks
Local	-15844.6	1911.7	637.3
Random	-28433.7	5423.3	1322.8
CCLB	-26075.2	4150.4	1286.4
ECLB	-24963.4	5878.9	1272.3
CACTE	**-4346.3**	**771.5**	**7**

Table 5. The impact of five mobile devices

Performance Metrics / Different Algorithms	The Total Reward	The Total Average Latency	The Total Number of Dropped Tasks
Local	-28107.9	3263.1	1197.1
Random	-45004.7	7113.3	2250.4
CCLB	-43969.1	6012.2	2265.7
ECLB	-36763.8	7395.3	1957.5
CACTE	**-16771**	**2922**	**599**

Table 6. The impact of six mobile devices

Performance Metrics / Different Algorithms	The Total Reward	The Total Average Latency	The Total Number of Dropped Tasks
Local	-50771.3	8564.0	2555.8
Random	-50771.3	8564.0	2555.8
CCLB	-45420.3	5151.1	2622.3
ECLB	-39509.1	8838.6	2044.7
CACTE	**-15262.0**	**2677.6**	**299.1**

7. Conclusions and future work

In this paper, we have mainly studied the multi-user collaborative task-execution problem under the limited computing, networks and battery capacity conditions in EH-D2D networks. To cope with this problem, we first designed a system model that included EH-D2D system architecture, an energy-queueing model and so on. The multi-user collaborative task-execution problem was then formulated. Finally, a CACTE scheme based on the MADDPG algorithm was developed, enabling multiple mobile devices to cooperate with one another to achieve a lower average delay in task processing and fewer dropped tasks. In comparison with four baseline algorithms (including Local, Random, ECLB and CCLB), our proposed CACTE scheme can achieve a higher total reward, lower total average delay and fewer dropped tasks. We also conducted comprehensive experiments to evaluate the performance of the CACTE scheme using different experimental parameters, such as task arrival rate, battery capacity, task workload, bandwidth and the number of mobile users. The experimental results demonstrated that the CACTE scheme can maximize the total reward and minimize the average delay and the number of dropped tasks. In conclusion, our work has provided useful guidelines which can be used to achieve collaborative task execution under conditions of limited computing, networks and battery capacity in mobile devices in EH-D2D networks.

Acknowledgments

This work was supported by the National Science Foundation of China (No. 61802095, 61572162, 61572251), the Zhejiang Provincial National Science Foundation of China (No. LQ19F020011, LQ17F020003), the Zhejiang Provincial Key Science and Technology Project Foundation (NO. 2018C01012), and the Open Foundation of State Key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications) (No. SKLNST-2019-2-15) and VC Research (VCR 0000111).

REFERENCES

[1] J. Wu, B. Cheng, M. Wang, and J. Chen, "Energy-efficient bandwidth aggregation for delay-constrained video over heterogeneous wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 1, pp. 30–49, 2016.

[2] J. Wu, B. Cheng, M. Wang, and J. Chen, "Quality-aware energy optimization in wireless video communication with multipath TCP," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2701–2718, 2017.

[3] Y. He, J. Ren, G. Yu, and Y. Cai, "D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks," *IEEE Trans. Wirel. Commun.*, vol. 18, no. 3, pp. 1750–1763, 2019.

[4] F. Jameel, Z. Hamid, F. Jabeen, S. Zeadally, and M. A. Javed, "A survey of device-to-device communications: Research issues and challenges," *IEEE Commun. Surv. Tutorials*, vol. 20, no. 3, pp. 2133–2168, 2018.

[5] M. K. Pedhadiya, R. K. Jha, and H. G. Bhatt, "Device to device communication: A survey," *J. Netw. Comput. Appl.*, vol. 129, pp. 71–89, 2019.

[6] H. Gao, C. Liu, Y. Li, and X. Yang, "V2VR: reliable hybrid-network-oriented V2V data transmission and routing considering RSUs and connectivity probability," *IEEE Trans. Intell. Transp. Syst.*, 2020.

[7] J. Liu, N. Kato, J. Ma, and N. Kadowaki, "Device-to-device communication in LTE-advanced networks: A survey," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 1923–1940, 2014.

[8] Z. H. Qian and X. Wang, "Reviews of D2D technology for 5G communication networks," *J. Commun.*, vol. 37, no. 7, pp. 1–14, 2016.

[9] A. Sultana, L. Zhao, and X. Fernando, "Efficient resource allocation in device-to-device communication using cognitive radio technology," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10024–10034, 2017.

[10] F. Tang, Z. M. Fadlullah, N. Kato, F. Ono, and R. Miura, "AC-POCA: Anticoordination game based partially overlapping channels assignment in combined UAV and D2D-based networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 2, pp. 1672–1683, 2018.

[11] Q. Yuan *et al.*, "Cross-Domain Resource Orchestration for the Edge-Computing-Enabled Smart Road," *IEEE Netw.*, vol. 34, no. 5, pp. 60–67, 2020.

[12] Q. Yuan, H. Zhou, J. Li, Z. Liu, F. Yang, and X. S. Shen, "Toward efficient content delivery for automated driving services: An edge computing solution," *IEEE Netw.*, vol. 32, no. 1, pp. 80–86, 2018.

[13] G. Luo *et al.*, "Software defined cooperative data sharing in edge computing assisted 5G-VANET," *IEEE Trans. Mob. Comput.*, 2019.

[14] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3887–3901, 2016.

[15] J. Feng, L. Zhao, J. Du, X. Chu, and F. R. Yu, "Computation offloading and resource allocation in D2D-enabled mobile edge computing," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[16] L. Huang, S. Bi, and Y. J. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mob. Comput.*, 2019.

[17] J. Zhang, J. Du, Y. Shen, and J. Wang, "Dynamic Computation Offloading with Energy Harvesting Devices: A Hybrid Decision Based Deep Reinforcement Learning Approach," *IEEE Internet Things J.*, 2020.

[18] B. He, S. Bi, H. Xing, and X. Lin, "Collaborative Computation Offloading in Wireless Powered Mobile-Edge Computing Systems," *arXiv Prepr. arXiv1908.09334*, 2019.

[19] H. Zhao, S. Deng, Z. Liu, J. Yin, and S. Dustdar, "Distributed redundancy scheduling for microservice-based applications at the edge," *IEEE Trans. Serv. Comput.*, 2020.

[20] J. He, D. Zhang, Y. Zhou, and Y. Zhang, "A truthful online mechanism for collaborative computation offloading in mobile edge computing," *IEEE Trans. Ind. Informatics*, vol. 16, no. 7, pp. 4832–4841, 2019.

[21] X. Chen, L. Pu, L. Gao, W. Wu, and D. Wu, "Exploiting massive D2D collaboration for energy-efficient mobile edge computing," *IEEE Wirel. Commun.*, vol. 24, no. 4, pp. 64–71, 2017.

[22] H. T. Nguyen, H. D. Tuan, T. Q. Duong, H. V. Poor, and W.-J. Hwang, "Joint D2D Assignment, Bandwidth and Power Allocation in Cognitive UAV-Enabled Networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 3, pp. 1084–1095, 2020.

[23] Y. Huang, Y. Liu, and F. Chen, "NOMA-aided mobile edge computing via user cooperation," *IEEE Trans. Commun.*, vol. 68, no. 4, pp. 2221–2235, 2020.

[24] M. Sun, X. Xu, Y. Huang, Q. Wu, X. Tao, and P. Zhang, "Resource Management for Computation Offloading in D2D-Aided Wireless Powered Mobile-Edge Computing Networks," *IEEE Internet Things J.*, 2020.

[25] F. Wang, J. Xu, and S. Cui, "Optimal Energy Allocation and Task Offloading Policy for Wireless Powered Mobile Edge Computing Systems," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 4, pp. 2443–2459, 2020.

[26] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," *arXiv Prepr. arXiv1703.06182*, 2017.

[27] J. Foerster *et al.*, "Stabilising experience replay for deep multi-agent reinforcement learning," *arXiv Prepr. arXiv1702.08887*, 2017.

[28] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in neural information processing systems*, 2017, pp. 6379–6390.

[29] G. Luo, H. Zhang, H. He, J. Li, and F.-Y. Wang, "Multi-agent Adversarial Collaborative Learning via Mean-Field Theory," *IEEE Trans. Cybern.*

[30] X. Yang, S. Zhou, and M. Cao, "An approach to alleviate the sparsity problem of hybrid collaborative filtering based recommendations: the product-attribute perspective from user reviews," *Mob. networks Appl.*, pp. 1–15, 2019.

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

[32] X. Qiu, L. Liu, W. Chen, Z. Hong, and Z. Zheng, "Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8050–8062, 2019.

[33] S. Deng *et al.*, "Dynamical resource allocation in edge for trustable Internet-of-Things systems: A reinforcement learning method," *IEEE Trans. Ind. Informatics*, vol. 16, no. 9, pp. 6103–6113, 2020.

[34] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, 2017.

[35] W. Jiang, G. Feng, S. Qin, and T. S. P. Yum, "Efficient D2D content caching using multi-agent reinforcement learning," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2018, pp. 511–516.

[36] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wirel. Commun.*, vol. 16, no. 8, pp. 4924–4938, 2017.