EURASIP Journal on Advances in Signal Processing

RESEARCH **Open Access**

A cloud-edge collaborative computing framework using potential games for space-air-ground integrated IoT

Yuhuai Peng[1*], Xiaoliang Guang[1], Xinyu Zhang[1], Lei Liu[2], Cemulige Wu[3] and Lei Huang[2]

*Correspondence: pengyuhuai@mail.neu.edu.cn

[1] School of Computer Science and Engineering, Northeastern University, No. 11, Lane 3, Culture Road, Heping District, Shenyang 110819, China
[2] The Guangzhou Institute of Technology, Xidian University, Huangpu District Zhongxin Knowledge City, Guangzhou 510555, China
[3] The Meta-Networking Research Center, the University of Electro-Communications, Tokyo 182-8585, Japan

Abstract
As a critical component of space-air-ground integrated IoT, the aerial network provides highly reliable, low-latency and ubiquitous information services to ground users by virtue of their high mobility, easy deployment and low cost. However, the current computation and resource management model of air-ground integrated networks are insufficient to meet the latency demanding of emerging intelligent services such as autonomous systems, extended reality and haptic feedback. To tackle these challenges, we propose a computation offloading and optimization method based on potential game. First, we construct an cloud-edge collaborative computing model. Secondly, we construct Offloading Decision Objective Functions (ODOF) with the objective of minimum task processing latency and energy consumption. ODOF is proved to be a Mixed Inferior Nonlinear Programming (MINLP) problem, which is hard to solve. ODOF is converted to be a full potential game, and the Nash equilibrium solution exists. Then, a computational resource allocation algorithm based on Karush–Kuhn–Tucker (KKT) conditions is proposed to solve resource allocation problem. On this basis, a distributed game-based computational offloading algorithm is proposed to minimize the offloading cost. Extensive simulation results demonstrate that the convergence performance of the proposed algorithm is reduced by 50%, the convergence time is reduced by 13.3% and the average task processing delay is reduced by 10%.

Keywords: Space-air-ground integrated IoT, Cloud-edge collaborative computing, Resource allocation, Offloading decisions, Potential game

1 Introduction

The space-air-ground integrated IoT is the integration network of interconnected devices, sensors and systems that can communicate and share data seamlessly across different dimensions, including space, air and ground [1, 2]. Relying on the long-endurance advantage of aerial platforms, drones, balloons and other air-based infrastructure, aerial network provides low-cost and wide-area coverage capability for 5 G and beyond (B5G/6 G) [3–5] with the massive deployment of high-performance computing devices in aerial platforms and the widespread application of air-to-ground communication (ATG) for B5G/6 G network and non-terrestrial networking (NTN) technology, and gradually forms informative and intelligent air-ground integrated ubiquitous network system [6, 7]. Constructing an cloud-edge collaborative computing model of air-ground integrated networks, orchestrating and controlling computing-storage-communication-perception resources to provide ubiquitous real-time service support for explosive growth of emerging services, such as holographic communication, extended reality, haptics, telemedicine and health monitoring.

However, with the booming development of B5G/6 G and the continuous influx of various emerging services, the heterogeneous ground terminals with limited capacity are finding it tough to meet the demand for high reliability and ultra-low latency services. The imbalance in the spatial and temporal distribution of computing, storage, communication and other system resources leads to inefficient task scheduling in the ground network, which makes it cumbersome to adapt accurately to varying task requirements [8]. In addition, the computational decision is subject to multiple constraints such as device computational capacity, available communication resources and channel selection, making it difficult to satisfy both latency and power requirements simultaneously [9].

To solve the above problems, the scholars in the field proposed to use aerial network-assisted communication-computation to alleviate the pressure of ground terminals [10, 11]. Heuristic and group intelligence methods achieve group offloading optimization by randomly or strategically changing individual behaviours, which have slow convergence speeds and are prone to falling into local optima [12]. Machine learning- and deep learning-based methods rely on a large amount of data for training and autonomous learning to obtain optimal offloading decisions [9, 13, 14]. These methods require the design of complex network models with high algorithmic complexity [15, 16]. Game-based methods [17, 18] simulate the offloading process by analysing the resource competition between devices to solve the optimal offloading decision. Its complexity is closely related to the amount of tasks.

Aiming at the problem of cloud-edge collaborative computing of air-ground integrated networks, this paper proposes a distributed computing offloading and resource allocation (DCORA) optimization scheme based on distributed gaming. The main contributions of this paper are as follows.

1. We construct an cloud-edge collaborative computing model. With the goal of minimizing task processing latency and energy consumption, an offloading decision objective function (ODOF) is constructed. The ODOF is proved to be a mixed inferior nonlinear programming problem (MINLP), it is hard to solve because of its nonlinear and non-convex nature. ODOF is converted to be a full potential game, and the Nash equilibrium solution exists.
2. The resource allocation subproblem is proven to be a convex optimization problem under specified available resources and task states. The resource allocation problem is converted into an unconstrained problem using Lagrange number multipliers. A Karush–Kuhn–Tucker (KKT) condition-based computational resource allocation (KCRA) algorithm is proposed to solve it. On this basis, DCORA algorithm is proposed to minimize the offloading cost by jointly optimizing the offloading mode selection, channel selection and offloading object selection.

3. Numerous simulation results show that DCORA has a fast convergence speed and can effectively reduce the task offloading delay and energy consumption. DCORA obtains minimum offload costs with changes in the number of channels, bandwidth, amount of task data and number of tasks. Compared with the traditional schemes, the convergence performance of DCORA is reduced by about 50%, the convergence time is shortened by about 13.3% and the average task processing delay is reduced by 10%.

The rest of the paper is organized as follows: In Sect. 2, related work is presented. In Sect. 3, the cloud-edge collaborative computing model and the objective problem are presented. In Sect. 4, the KCRA algorithm and the DCORA algorithm are described in detail. Finally, we conduct simulation experiments and results analysis in Sect. 5, and conclusions are drawn in Sect. 6.

2 Related work

This subsection provides an introduction to mainstream computational offloading algorithm research efforts, including heuristic and swarm intelligence methods, machine learning (ML) and deep learning (DL) approaches and game-based strategies.

Swarm intelligence algorithms are stochastic algorithms inspired by biological or physical phenomena, where individuals continuously modify their actions to achieve collective optimization of the population. These algorithms have been widely used to address multi-objective optimization problems, such as computation offloading and resource allocation. Lv et al. [19] proposed an approach based on heuristic algorithms to predict the impact of offloading decisions. Ali et al. [20] introduced an optimization model based on discrete non-dominated sorting genetic algorithm to handle discrete multi-objective task scheduling problems. Dai et al. [21] proposed an optimization method based on particle swarm optimization by splitting tasks for offloading. Dong et al. [22] combined particle swarm optimization and quantum particle swarm optimization to propose a computation offloading task strategy and validated its effectiveness. Yuan et al. [23] established a fine-grained task offloading model, proposing a task prediction algorithm based on the long short-term memory neural network model and an online offloading algorithm based on particle swarm optimization. When the number of offloading tasks is small, swarm intelligence algorithms exhibit higher complexity. However, with a larger number of tasks, swarm intelligence algorithm often encounters challenges such as prolonged convergence time and being trapped in local optima.

Computation offloading methods based on ML or DL, with neural networks at their core, rely on extensive data for training and autonomous learning, can address various complex environments. Yang et al. [24] devised a collaborative offloading and resource allocation scheme based on energy prediction, optimizing transmission power, computing resource allocation and task offloading ratios. Liu et al. [25] proposed a distributed optimization problem for offloading parameters and designed strategies for task offloading and energy conservation response. Gao et al. [26] presented a multi-tier fog computing system that predicts offloading and resource allocation, reducing average power consumption. Dai et al. [27, 28] employed Lyapunov optimization to transform stochastic problems into deterministic ones for each time slot, using an asynchronous behaviour critic algorithm to find the optimal offloading strategy. Liu et al. [29] constructed a mathematical optimization model for power consumption and time overhead, proposing intelligent task offloading solutions, enhancing system network performance through decision tree algorithms and double-depth algorithms. Qu et al. [30] combined multiple parallel deep neural networks with Q-learning to derive optimal offloading strategies from dynamic environments. However, methods based on ML or DL currently face challenges such as slow learning speed, weak adaptability to new environments and reliance on prior data for training.

Yu et al. [31] proposed a mixed-strategy Nash equilibrium (NE) based on virtual game theory, decomposing the offloading decision problem of the entire system into a hierarchical game problem. Xu et al. [32] designed a fuzzy task offloading and resource allocation scheme based on Takagi–Sugeno fuzzy neural networks and game theory to minimize user task processing latency. Wang et al. [33] introduced a resource allocation incentive mechanism based on Stackelberg game and devised an optimization strategy using the alternating direction method of multipliers. Pham et al. [34] utilized precise potential game theory to design a low-complexity distributed offloading scheme and determined the optimal offloading ratio and resource allocation using the subgradient method. Luo et al. [35] proposed a distributed offloading decision algorithm based on game theory models, achieving Nash equilibrium through self-learning to minimize offloading latency and cost. Huang et al. [36] addressed the latency optimization offloading problem using non-cooperative game theory and provided a solution. Teymoori et al. [37] described the offloading decision process as a stochastic game model to minimize mutual interference during channel access, solving for Nash equilibrium based on multi-agent reinforcement learning. Mensah et al. [38] combined device-to-device (D2D) communication with vehicular networks, formulating the task offloading and resource allocation problem as a mixed-strategy game and solving for Nash equilibrium. Yang et al. [39] regarded the computation offloading process as a competitive game to minimize the cost of executing a single task, proposing a lightweight algorithm to solve for Nash equilibrium. Fan et al. [40] presented an offloading scheme based on non-cooperative game theory to alleviate node load, balance task delays and demonstrated the existence of Nash equilibrium using variational inequalities and regularization techniques. Pham et al. [41], based on coalition game theory, studied a low-complexity algorithm that guarantees convergence, compared it with three baseline schemes and verified its effectiveness.

Compared to methods based on swarm intelligence, ML or DL, game-based approaches are more suitable for distributed computing due to their lower complexity and faster convergence. Therefore, this paper proposes a distributed game-based computation offloading optimization method to address the problem more efficiently.

3 System model

3.1 Cloud-edge collaborative computing model

The air-ground integrated IoT network model is shown in Fig. 1, and it includes a ground network, a low-altitude network and a high-altitude network. The ground network consists of emergency communication vehicles, IoT devices, base stations, intelligent robots and other devices. The low-altitude part of the network consists of

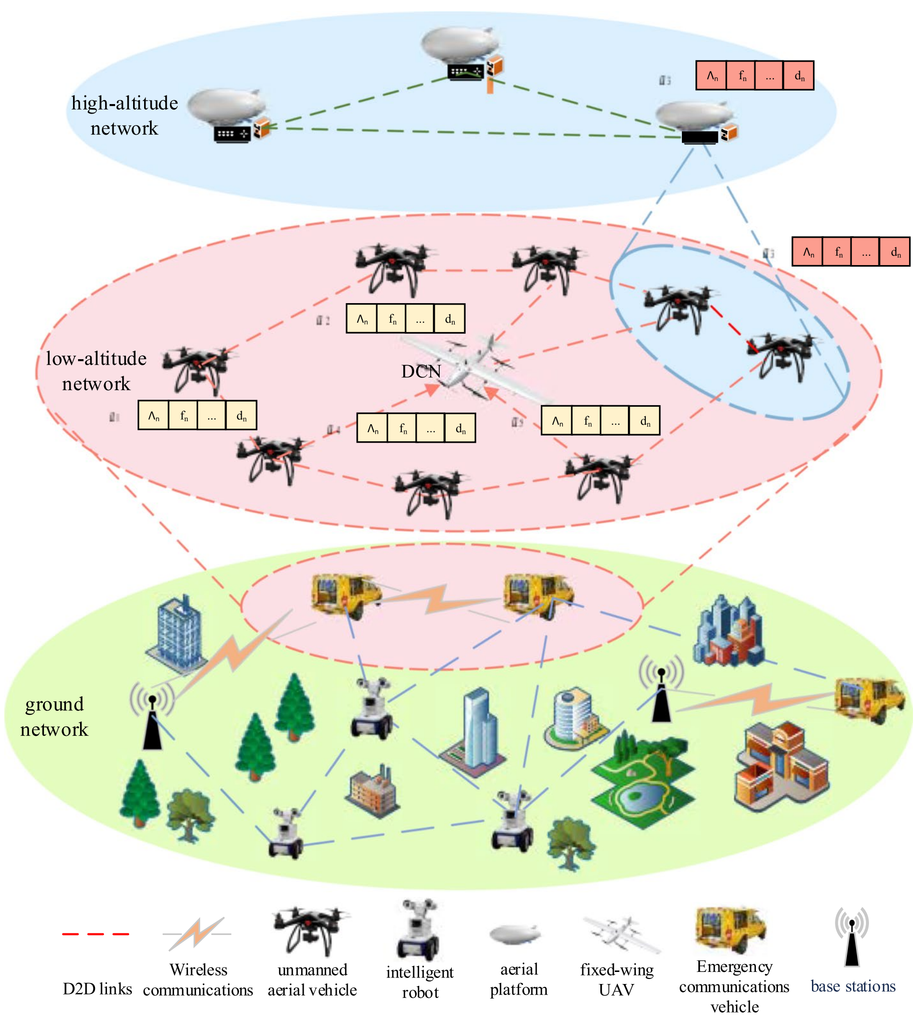

Fig. 1 The cloud-edge collaborative computing model

UAV swarms and fixed-wing UAV. The fixed-wing UAV with the strongest computational capacity is called distributed computing node (DCN). Since small UAVs have limited computing and communication capabilities, the fixed-wing UAV is provided to support the UAV swarm. The high-altitude network consists of floating airships with deployed mobile edge computing (MEC).

The cloud consists of floating airships with deployed mobile edge computing (MEC), and edge consists of UAVs and fixed-wing UAV. The UAV swarms in the low-altitude network will assist the ground equipment with computing tasks. The UAVs are divided into two disjoint sets based on their current operational status. UAVs with computing tasks are referred to as active devices, denoted by set $\mathcal{N} = \{1, 2, \ldots, N\}$. UAVs without computing tasks are referred to as auxiliary devices, denoted by set $\mathcal{H} = \{1, 2, \ldots, H\}$. The DCN establishes a D2D link and jointly computes with the devices in $\mathcal{H}$. It will help the UAV terminal to generate offload decisions and perform mission computation. If the UAV is not connected to the DCN, and the UAV is within the communication range of the aerial platform. The aerial platform in the high-altitude segment will take over some of the computing tasks from the UAV according to the offload strategy. The available bandwidth B is divided into K_{d+e} mutually orthogonal subcarriers, including K_d for D2D communication and K_e for cellular communication, which can be represented as follows: $\mathcal{K} = \{1, 2, \ldots, K_d, K_{d+1}, \ldots, K_{d+e}\}$.

The state information of computing task $n \in \mathcal{N}$ can be described by set $\Lambda_n = \{s_n, c_n\}$. s_n is the data size of the task and measured in bits. c_n is the required computing resource to compute one bit of the task and measured in CPU cycles per bit. We use a binary offloading policy for offloading tasks that are offloaded completely without task segmentation. The offloading mode variate $x_{n,m}$ represents whether the task n is offloaded to the destination m, where $x_{n,m} \in \{0, 1\}$, $m \in \mathcal{H} \cup \{0\}$ represents the offloading destination of the device n, 0 is the edge server identity. In the three different offloading modes, the practical significance of $x_{n,m}$ is expressed as follows:

1. When $x_{n,m} = 1, m = n$, it means that task n executes locally.
2. When $x_{n,m} = 1, m \in \mathcal{N}$, it means that the task n is offloaded to DCNs for execution via the D2D link.
3. When $x_{n,m} = 1, m = 0$, it means that the task n is offloaded to MEC for execution via the cellular link.

3.2 Delay and energy model

(1) *Local computing model* When the task n is executed locally, the task does not require transmission, so the offloading cost only includes the computation delay $t_n^{loc,cmp}$ and energy consumption $e_n^{loc,cmp}$.

$$t_n^{loc,cmp} = \frac{s_n c_n}{f_n} \tag{1}$$

$$e_n^{loc,cmp} = \kappa_n c_n f_n^2 \tag{2}$$

Where f_n is the number of CPU cycles per second of terminal n, which represents its computing power. κ_n denotes the effective switching capacity of terminal n, and its value is a constant term. In order to prolong the usage time of the terminal device and consider the energy consumption and delay during the task completion process, we sets the local offloading cost function is v_n^{loc}.

$$v_n^{loc} = \beta_t t_n^{loc,cmp} + \beta_e e_n^{loc,cmp} \tag{3}$$

Where $\beta_t, \beta_e \in [0, 1]$ and $\beta_t + \beta_e = 1$ are weight constants. β_t denotes the weight of delay in the completion of task, and β_e denotes weight of the energy consumption of device.

(2) *D2D offloading* When task n is offloaded to DCNs via D2D link, the uplink transmission rate $r_{n,h}$ can be found in Eq. 4.

$$r_{n,h} = \bar{B} log_2 \left(1 + \frac{x_{n,h} w_{n,h}^k p_n h_{n,h}^k}{N_0 + \sum_{i=1}^{N} x_{i,h} w_{i,h}^k p_i h_{i,h}^k} \right) \tag{4}$$

Where $\bar{B} = B/k_{d+e}$ is the sub-carrier bandwidth. $k \in \{1, 2, \ldots, K_d\}$ represents the exclusive channel of D2D communication mode. A value of 1 for the binary variable

$w_{n,h}^{k} = \{0, 1\}$ means that the channel k is selected when the task n is offloaded to the destination h, and $h \in \mathcal{H}$. p_n denotes the transmission power of the device. $h_{n,h}^{k}$ represents the channel gain during D2D transmission, the value is related to the transmission distance and the maximum D2D transmission distance is d_{max}. N_0 is the noise power within each sub-channel. For efficient use of spectrum resource, devices in the same transmission mode can be allowed to be reused in the sub-channels during communication, $\sum_{i=1}^{N} x_{i,h} w_{i,h}^{k} p_i h_{i,h}^{k}$ denotes interference between D2D links due to spectral reuse.

Therefore, the computation delay $t_n^{d2d,cmp}$ in D2D offloading mode can be found in Eq. 5.

$$t_n^{d2d,cmp} = \frac{s_n c_n}{f_{n,h}} \tag{5}$$

Where $f_{n,h}$ represents the computing resource that can be allocated when the task n is offloaded to the DCN m. The delay and energy consumption during transmission can be expressed as $t_n^{d2d,trans}$ and $e_n^{d2d,trans}$, respectively.

$$t_n^{d2d,trans} = \frac{s_n}{r_{n,h}} \tag{6}$$

$$e_n^{d2d,trans} = p_n t_n^{d2d,trans} \tag{7}$$

Considering that the offloading result response signal data size is much less than the task data size, the result backhaul delay is ignored. Therefore, the D2D offloading cost function v_n^{d2d} is defined as the weighted sum of computing delay, transmission delay and transmission energy consumption.

$$v_n^{d2d} = \beta_t (t_n^{d2d,cmp} + t_n^{d2d,trans}) + \beta_e e_n^{d2d,trans} \tag{8}$$

(3) *Edge offloading* When task n is offloaded to MEC via cellular link, the uplink transmission rate $r_{n,0}$ can be defined as Eq. 9.

$$r_{n,0} = \bar{B} log_2 \left(1 + \frac{x_{n,0} w_{n,0}^{k} p_n h_{n,0}^{k}}{N_0 + \sum_{i=1}^{N} x_{i,0} w_{i,0}^{k} p_i h_{i,0}^{k}} \right) \tag{9}$$

Where $k \in \{k_{d+1}, \ldots, K_{d+e}\}$ represents the exclusive channel of cellular communication mode. The binary variable $w_{n,0}^{k} = \{0, 1\}$ value of 1 indicates that the channel k is selected when the task n is offloaded to MEC. $h_{i,0}^{k}$ represents the channel gain during cellular transmission. Devices in the same transmission mode are also allowed to be reused to sub-channels during communication, $\sum_{i=1}^{N} x_{i,0} w_{i,0}^{k} p_i h_{i,0}^{k}$ represents interference due to spectral reuse between cellular links.

From the previous discussion, the task computation delay $t_n^{mec,cmp}$ can be found in Eq. 10.

$$t_n^{mec,cmp} = \frac{s_n c_n}{f_{n,0}} \tag{10}$$

Where $f_{n,0}$ represents the computing resource allocated when the task n is offloaded to MEC. The delay of the task transmission process and energy consumption can be expressed as $t_n^{mec,trans}$ and $e_n^{mec,trans}$, respectively.

$$t_n^{mec,trans} = \frac{s_n}{r_{n,0}} \tag{11}$$

$$e_n^{mec,trans} = p_n t_n^{mec,trans} \tag{12}$$

In summary, the MEC offloading cost function can be expressed as v_n^{mec}.

$$v_n^{mec} = \beta_t (t_n^{mec,cmp} + t_n^{mec,trans}) + \beta_e e_n^{mec,trans} \tag{13}$$

(4) *Problem formulation* Compared with the edge computing model, D2D offloading can ensure shorter communication delay to reduce the upstream network transmission pressure. On the other hand, an edge server can provide a significant amount of computational resources. The aim of our work is to realize the target problem joint optimization and minimize the global task offloading cost through the offloading mode, offloading destination selection, channel selection and computing resource allocation. The current moment task n cost function and global target optimization function **P1** can be found in Eqs. 14 and 15, respectively.

$$v_n = x_{n,0} v_n^{mec} + x_{n,h} v_n^{d2d} + x_{n,n} v_n^{loc} \tag{14}$$

$$\begin{aligned} &\mathbf{P1} : \min_{a_n} \sum_{n=1}^{N} v_n \\ s.t. \quad & C1 : x_{n,m} \in \{0,1\} \\ & C2 : x_{n,0} + x_{n,h} + x_{n,n} = 1 \\ & C3 : 0 \leq f_{n,0} \leq f_0 \\ & C4 : 0 \leq f_{n,h} \leq f_h \\ & C5 : 0 \leq \sum_{n=1}^{N} x_{n,0} f_0 = f_0 \\ & C6 : 0 \leq \sum_{n=1}^{N} x_{n,h} f_h = f_h \\ & C7 : \beta_t, \beta_e \in [0,1], \beta_t + \beta_e = 1 \end{aligned} \tag{15}$$

Where $a_n = \left\{x_{n,m}, w_{n,m}^k, f_{n,0}, f_{n,h}\right\}$ is the set of variables to be optimized. *C1* states that the offload mode decision is a binary value and each task must and can choose only one offload mode. *C2* and *C3* indicate that the computational resources allocated to the tasks when offloading to the MEC and DCN must be within the maximum constraint. *C4* and *C5* indicate that all tasks offloaded to the MEC and DCN together occupy the entire computational resources. *C6* specifies the range of value for the weights of the cost function.

(5) *Game theory analysis* The objective optimization problem can be described as a multi-knapsack problem. The multi-knapsack problem refers to the task of skillfully selecting a subset of items from a finite set, each with specific weights or profits, and efficiently placing them into a limited capacity knapsack. The objective is to maximize or minimize the total weight or total profit of the items loaded into the knapsack [42]. Specifically, limited computing tasks n are equivalent to items, the offloading destination

m is the backpack, total profit is the weighted sum of delay and energy consumption. The target problem is the reasonable decision action $a_n = \{x_{n,m}, w_{n,m}^k, f_{n,0}, f_{n,h}\}$ to minimize the total profit of the backpack. The multi-knapsack problem has been widely shown to be an NP-hard problem [43]. Since the decision variables $\{x_{n,m}, w_{n,m}^k\}$ are discrete variables, $\{f_{n,0}, f_{n,h}\}$ are continuous values in a finite interval, and the objective function has its nonlinear properties. Thus, the problem **P1** is a mixed integer nonlinear programming (MINLP). In order to solve this problem, a distributed computation offloading and resource allocation scheme is formulated.

Game theory is a mathematical method for analysing decision problems. When there is a competition for a certain resource among multiple participants, the decisions between participants often influence each other. Game theory studies the decision process of participants through the influence relationship [36].

During the game process, all devices share global resource and state information. The goal of the decision is to continuously minimize the offloading cost function, in order to obtain the optimal offloading experience. The game can be expressed as follows: $G =< \mathcal{N}, \mathcal{A}, \mathcal{U} >$. $\mathcal{N}$ is a set of players as well as a set of tasks. $\mathcal{A} = \{d_n | n \in \mathcal{N}\}$ is the set of actions that the player n can select. $d_n = \{x_{n,m}, w_{n,m}^k\}$ represents the action decision made by player n regarding the selection of the offloading destination and channel. $\mathcal{U} = \{u_n(d_n, d_{-n}) | n \in \mathcal{N}\}$ represents the player's utility function value under the current action. The utility function is set to Eq. 16, each player takes action targeting lowering the cost of global offloading.

$$u_n(d_n, d_{-n}) = v_n(d_n, d_{-n}) + \sum_{i \neq n} [v_i(d_i, d_{-i}) - v_i(d_i, d_{-i\backslash n})] \tag{16}$$

Where d_{-n} is the vector of the other player's current actions. $v_i(d_i, d_{-i\backslash n})$ represents the utility function value of player i when player n gives up switching action. $\sum_{i \neq n} [v_i(d_i, d_{-i}) - v_i(d_i, d_{-i\backslash n})$ illustrates the sum of the value of the other player utility function that changes when the player n switches its action. Each player takes action d_n^* based on the action combination d_{-n}^* chosen by the other players. Game G reaches a Nash equilibrium when Eq. 17 is satisfied [33].

$$u_n(d_n^*, d_{-n}^*) \leq u_n(d_n, d_{-n}^*) \qquad \forall n \in \mathcal{N}, \forall d_n \in \mathcal{A} \tag{17}$$

Where $d_{-\boldsymbol{n}}^* = \left(d_1^*, \ldots, d_{n-1}^*, d_{n+1}^*, \ldots, d_N^*\right)$. Each player has already chosen their optimal action. Furthermore, it is important to note that no player will alter their decision. Specifically, any player who chooses an action other than d_n^* will not be able to achieve a lower utility value.As a result, the current set of actions represents the global optimum.

Definition If a game has a potential function $P_n(d_n, d_{-n})$ such that Eq. 18 holds for all $\forall n \in \mathcal{N}, \forall d_n \in \mathcal{A}$, then the game is referred to as a completely potential game [34].

$$u_n(\bar{d}_n, d_{-n}) - u_n(d_n, d_{-n}) = P_n(\bar{d}_n, d_{-n}) - P_n(d_n, d_{-n}) \tag{18}$$

Completely potential games are a specific type of game. What sets them apart from ordinary games is that when a player unilaterally changes their own action, the potential function accurately reflects changes in the player's utility function.

1 ***Proof*** $G =< \mathcal{N}, \mathcal{A}, \mathcal{U} >$ is a completely potential game.

$$\begin{aligned}
& u_n(\overline{d_n}, d_{-n}) - u_n(d_n, d_{-n}) \\
&\quad = v_n(\overline{d_n}, d_{-n}) + \sum_{i \neq n}[v_i(d_i, \overline{d_{-i}}) - v_i(d_i, \overline{d_{-i\backslash n}})] \\
&\qquad - v_n(d_n, d_{-n}) + \Big\{ \sum_{i \neq n}[v_i(d_i, d_{-i}) - v_i(d_i, d_{-i\backslash n})]\Big\} \\
&\quad = v_n(\overline{d_n}, d_{-n}) - v_n(d_n, d_{-n}) + \sum_{i \neq n}[v_i(d_i, \overline{d_{-i}}) \\
&\qquad - v_i(d_i, d_{-i})] - \sum_{i \neq n}[v_i(d_i, \overline{d_{-i\backslash n}}) - v_i(d_i, d_{-i\backslash n})] \\
&\quad = v_n(\overline{d_n}, d_{-n}) - v_n(d_n, d_{-n}) \\
&\qquad + \sum_{i \neq n}[v_i(d_i, \overline{d_{-i}}) - v_i(d_i, d_{-i})] \\
&\quad = \sum_{i=1}^{N}[v_i(\overline{d_i}, d_{-i}) - v_i(d_i, d_{-i})] \\
&\quad = \sum_{i=1}^{N} v_i(\overline{d_i}, d_{-i}) - \sum_{i=1}^{N} v_i(d_i, d_{-i}) \\
&\quad = P_n(\overline{d_n}, d_{-n}) - P_n(d_n, d_{-n})
\end{aligned} \tag{19}$$

The player's switch action from d_n to $\bar{d_n}$ will cause a change in the utility value of other players. But when the player n gives up switching decision and other users maintain their decision, the other players' function utility value remains the same $v_i(d_i, \overline{d_{-i\backslash n}}) = v_i(d_i, d_{-i\backslash n})$. Consequently, during each iteration, there is always a potential function $P_n(d_n, d-n) = \sum_{i=1}^{N} v_i(d_i, d_{-i})$ regardless of how the player's n action changes.

Theorem 1 *All potential games with finite action spaces must have a Nash equilibrium.*

It is easy to verify that action space $\mathcal{A} = \{d_n | n \in \mathcal{N}\}$ is finitely closed set. In conclusion, the game $G =< \mathcal{N}, \mathcal{A}, \mathcal{U} >$ is a completely potential game whose Nash equilibrium is always exist.

4 Distributed computation offloading and resource allocation scheme

4.1 Computing resource allocation problem analysis

In the context of computational offloading, UAVs, MEC and DCNs serve as auxiliary nodes. When computational resources are allocated to MEC, assuming that the set of I tasks unloaded to MEC for calculation is I_0, and the channel selection is fixed. The computing resource assigned to each computation task can be defined as $\{f_i | i \in I_0\}$.

Therefore, the subproblem function associated to computing resource allocation in problem **P1** can be converted to **P2**.

$$\begin{aligned} \mathbf{P2}: &\min_{f_i} F(f_i) = \min_{f_i} \sum_{i \in I_0} \frac{s_i c_i}{f_i} \\ s.t. \quad & C1: 0 \le f_i \le f_0 \\ & C2: \sum_{i \in I_0} f_i = f_0 \end{aligned} \tag{20}$$

Theorem 2 *$F(f_i)$ is a strictly convex function.*

The Hessian matrix of the target problem **P2** *can be calculated by Eq.* 21.

$$\mathrm{H} = \begin{bmatrix} \frac{\partial^2 \mathrm{F}(\mathrm{f}_1)}{\partial (\mathrm{f}_1)^2} & \cdots & \frac{\partial^2 \mathrm{F}(\mathrm{f}_1)}{\partial (\mathrm{f}_1)\partial (\mathrm{f}_\mathrm{I})} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 \mathrm{F}(\mathrm{f}_\mathrm{I})}{\partial (\mathrm{f}_\mathrm{I})\partial (\mathrm{f}_1)} & \cdots & \frac{\partial^2 \mathrm{F}(\mathrm{f}_\mathrm{I})}{\partial (\mathrm{f}_\mathrm{I})^2} \end{bmatrix} \tag{21}$$

The second-order derivatives and mixed partial derivatives of $F(f_i)$ are given as follows:

$$\frac{\partial^2 F(f_i)}{\partial (f_i)^2} = \frac{2 s_i c_i}{f_i^3} \qquad \forall i \in I_0 \tag{22}$$

$$\frac{\partial^2 F(f_i)}{\partial (f_i) \partial (f_j)} = 0 \qquad \forall i \in I_0, i \neq j \tag{23}$$

Thus, the eigenvalues of the Hessian matrix H are all greater than zero, which means that the Hessian matrix of the target problem **P2** is positive definite. Moreover, the **P2** is a strictly convex function. KKT conditions can transform the optimization problem of equality and inequality mixed constraints into an unconstrained problem, which is a necessary and sufficient condition to judge that a certain point of convex programming is an extreme point. Consequently, **P2** can be solved by the Lagrange multiplier method with the KKT condition. The Lagrangian is constructed as described in Eq. 24.

$$\begin{aligned} &F(f_1, \ldots, f_I, \lambda, \mu_1, \ldots, \mu_I) \\ &\quad = \sum_{i \in I_0} \frac{s_i c_i}{f_i} + \lambda (\sum_{i \in I_0} f_i - f_0) + \sum_{i \in I_0} \mu_i (f_i - f_0) \end{aligned} \tag{24}$$

Where λ and $\mu_1, \ldots, \mu_I$ are non-negative Lagrange multiplier whose value changes with the iteration. The KKT condition can be calculated as follows:

$$-\frac{s_i c_i}{f_i^2} + \lambda + \mu_i = 0 \tag{25a}$$

$$\sum_{i \in I_0} f_i - f_0 = 0 \tag{25b}$$

$$\mu_i (f_i - f_0) = 0 \tag{25c}$$

During each iteration, the Lagrange multipliers can be updated according to a specific step-size formula to gradually approach the optimal solution. The Lagrange multiplier must greater than zero and cannot be zero when used as a denominator. Therefore, when the multipliers take negative values, we update them to an infinitesimal value ε_1. The update formula is as follows:

$$\lambda(t+1) = \max\{\varepsilon_1, \lambda(t) + \delta_1(t) \sum_{i \in I_0} (f_i - f_0)\} \tag{26a}$$

$$\mu_i(t+1) = \max\{\varepsilon_1, \mu_i(t) + \delta_1(t)(f_i - f_0)\} \tag{26b}$$

The resource allocated for each computation task can be expressed as follows: $\{f_i | i \in I_h\}$. The KKT conditions and the update formula for the Lagrange multipliers are as follows:

$$-\frac{s_i c_i}{f_i^2} + \lambda + \mu_i = 0 \tag{27a}$$

$$\sum_{i \in I_h} f_i - f_h = 0 \tag{27b}$$

$$\mu_i (f_i - f_h) = 0 \tag{27c}$$

$$\lambda(t+1) = \max\left\{\varepsilon_1, \lambda(t) + \delta_2(t) \sum_{i \in I_h} (f_i - f_h)\right\} \tag{28a}$$

$$\mu_i(t+1) = \max\left\{\varepsilon_1, \mu_i(t) + \delta_2(t)(f_i - f_h)\right\} \tag{28b}$$

We propose a KKT-based resource allocation algorithm as shown in Algorithm 1 to solve the target problem **P2**. Initially, the parameters are initialized, and then, the resource allocation decisions are computed based on the Lagrange multipliers. During the iterative process, we update the Lagrange multiplier values for the next iteration according to the step size. This process continues until the gap between the computed resources and node computing capacity is less than an infinitesimal value ε_2.

Algorithm 1 KCRA algorithm.

```
Input: $f_0, f_h, \lambda, \mu_i, \delta_1(t), \delta_2(t)$, Minimal threshold, the device set for selecting each
    node $I_0, I_h$.
Output: $\{f_i, \forall i \in I_0\}, \{f_i, \forall i \in I_h\}$
1:  if $Flag_{KCRA} = 1$ then
2:      while $|\sum_{i \in I_0}(f_i - f_0)| \geq \varepsilon_2$ do
3:          calculate the computing resource $f_i$ which is assigned to task in the current
    iteration according to Eq. 25a.
4:          Update the lagrange multiplier $\lambda, \mu_i$ according to Eq. 26a and Eq. 26b.
5:      end while
6:  else
7:      while $|\sum_{i \in I_h}(f_i - f_h)| \geq \varepsilon_2$ do
8:          calculate the computing resource $f_i$ which is assigned to task in the current
    iteration according to Eq. 27a.
9:          Update the lagrange multiplier $\lambda, \mu_i$ according to Eq. 28a and Eq. 28a.
10:     end while
11: end if
```

4.2 DCORA algorithm

The computation offloading based on distributed game algorithm is used to solve problem **P1**. Specifically, we utilize the finite improvement property of games to progressively eliminate dominated strategies, seeking a global optimum that satisfies Nash equilibrium. The UAV and MEC engage in collaborative decision-making through multiple rounds of information sharing. This information exchange encompasses task status, global channel conditions and computational resource status. The information exchange process for each round of iteration in this algorithm is detailed below.

(1) The UAV transmits the task status information and initial offloading decisions to the MEC as pilot signals.
(2) MEC collects all task status information, decision information and resource status information as a feedback signal. Then, MEC broadcasts it to all UAV to support decision update.
(3) The UAV receives the feedback information and decides whether to update the current decision according to Algorithm 2 and strive for renewal opportunities. To avoid redundant and ineffective computations, each device will exclude the decision made in the previous iteration from its available action space in each round of iteration.
(4) MEC collects update applications. To avoid local optimum, MEC randomly selects one device from the updated device set for decision update. And it broadcasts information as described in (2).

Repeat the above steps until no device applies for update, the specific algorithm is shown below.

Algorithm 2 DCORA algorithm.

```
Input: simulation parameters such as network topology, $\mathcal{N}, B, K_{d+e}, p_n, \beta_t, \beta_e, UAV'$ action space (available DCNs and channels in the range) and random initial action, etc.
Output: offloading decision $a_n = \{x_{n,m}, w_{n,m}^k\}$.
 if $Flag_{DCORA} = true$ then
     while flag do
         Calculate the current computing resource allocation based on Algorithm 1;
         Calculate $u_n(d_n, d_{-n})$ base on the decision of the current iteration;
         Establish the device update application set $UD = \emptyset$;
         for $n < \mathcal{N}$ do
             Establish the available destination decision set $AD$ and better decision storage collection $BD = \emptyset$ of UAV $i$;
             for $j < lenght(AD)$ do
                 Establish the available channel set AC;
                 Calculate the current computing resource allocation based on Algorithm 1;
                 for $k <$ lenght$(AC)$ do
                     Calculate $u'_n(d_n, d_{-n})$.
                     if $u'_n(d_n, d_{-n}) < u_n(d_n, d_{-n})$ then
                         Store the current decision $d_n^{'}$ to BD;
                     end if
                 end for
             end for
             if $BD \neq \emptyset$ then
                 Store the decision $d_n = rand(\{d'_n | d'_n \in BD\})$ into set UD;
             end if
         end for
         if $UD \neq \emptyset$ then
             Randomly select a device from set $UD$ for updating offloading decision;
         else
             flag = false;
         end if
     end while
 end if
```

During the running of our distributed offloading algorithm, each device parallel runs the algorithm 7–20 lines. The calculation complexity of the two cycles within the devices is $O(a * b)$. Where a is the length of available destination decision set, which value is $1 + H$. Moreover, b is the length of available channel set, which value is $K_{d+e} - 1$. Obviously, the computation complexity of DCORA is of $O(n^2)$. If the centralized decision-making is adopted, all devices offload the task information to the MEC. It makes the iterative decision, the corresponding computation complexity is $O(a * b * n)$, which is of $O(n^3)$.

5 Simulation and result analysis

5.1 Parameter setting

Consider a cellular network covering a range of 200×200 meters with $N = 20$ computation tasks and $H = 8$ MEC nodes. Each UAV can cover an area with a radius of $50m$ by D2D communication. Figure 2 illustrates a schematic diagram of the network

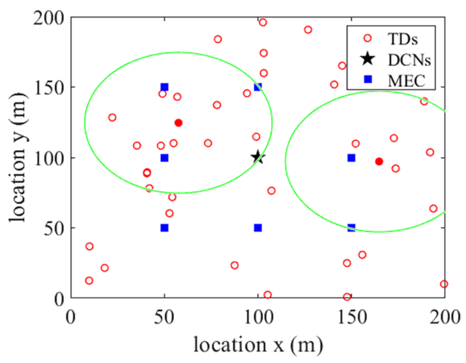

Fig. 2 Network topology simulation diagram

Table 1 Parameter setting

Parameter	Value
Bandwidth of channels	35 MHz
Channel gain of D2D	$128.1 + 37.6log_{10}d_{n,m}$
Channel gain of MEC	$148.1 + 40log_{10}d_{n,m}$
Number of channel of D2D,MEC	3,4
Channel noise	10^{-10} mw
Weight value of time and energy	0.5
Effective switching capacity of the device	10^{-27}
Device transmission power	500 mw
Data size	[0.2, 2] Mbits
Required number of CPU cycles	[15002000] cycles/bit
The computation capacity of device	[0.52]×10^9 cycles/s
The computation capacity of DCN	[6 10]×10^9 cycles/s
The computation capacity of MEC	40×10^9 cycles/s

topology. The channel number of cellular link and D2D link is $K_d = 3$ and $K_e = 4$, respectively. The total channel bandwidth is 35 MHz. Meanwhile, the channel gain is $128.1 + 37.6log_{10}d_{n,m}$ and $148.1 + 40log_{10}d_{n,m}$, respectively. Where $d_{n,m}$ represents the distance between the UAV and the offloading destination. And the channel noise is $N_0 = 10^{-10}$ mw. The task data size follows randomly distributed within [0.2, 2] Mbits. Furthermore, the calculation required number follows randomly distributed within [1500, 2000] cycle/bit. Table 1 lists the specific parameter [44–46].

5.2 Simulation and result analysis

The proposed scheme is discussed in comparison with the following four offloading schemes:

(1) Full local computing scheme.

(2) Full edge offloading scheme. In this case, all tasks will be offloaded to the edge server. The scheme includes channel allocation and the computing resource allocation process based on KCRA algorithm.
(3) Full random offloading scheme. In this scenario, the devices randomly select the offloading destination and channel, moreover allocate computing resource according to KCRA algorithm.
(4) DMCTO scheme [42]. In this scheme, the tasks offloaded to the DCN will be allocated computational resources evenly. We take the average of multiple runs as the final result.

Figure 3a illustrates the impact of the number of iterations on the global offloading cost. As the number of iterations increases, the global cost for different task quantities shows a decreasing trend and converges to a stable value. The stable value corresponds to the Nash equilibrium solution, which is the optimal offloading decision. It can be observed that as the number of tasks N increases, the global offloading cost also increases. The reason is that as the number of terminal devices increases, more tasks need to be computed, resulting in higher total computation time and energy consumption. Therefore, the cost of offloading increases with the number of devices. Figure 3b depicts how the number of devices affects the number of iteration times. It can be seen that as the number of tasks increases, the iteration times for task convergence also increase. The reason for this is that the greater the number of devices, the greater the number of feasible solutions. This means that DCORA needs to perform more iterations to find the optimal or suboptimal solution from a large number of feasible solutions. In summary, as the number of iterations increases, the DCORA algorithm gradually reduces the global cost and slows down the convergence speed. Compared to the DMCTO algorithm, the convergence performance has improved by approximately 50%, and the convergence time has been reduced by approximately 13.3%.

Figure 4 illustrates the impact of weights on average delay and energy consumption. With β increases, the average delay gradually reduced, while the average energy consumption gradually increases. As β varies from 0 to 1, the DCORA algorithm exhibits significantly lower average delay compared to the DMCTO algorithm. Thus, there

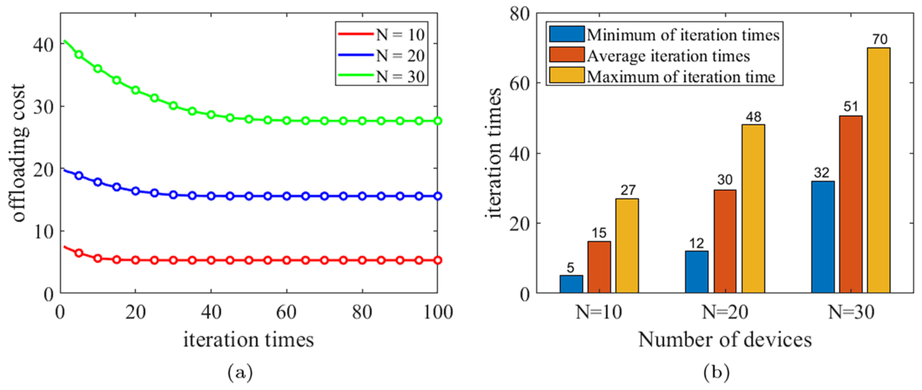

Fig. 3 **a** Relationship between iteration times and offloading cost with different tasks in DCORA and **b** relationship between the number of devices and iteration times in DCORA

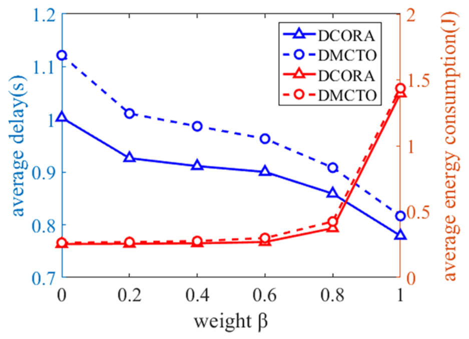

Fig. 4 The average processing delay and energy consumption change under different weights

is little difference in average energy consumption compared to the DMCTO algorithm. In summary, compared to the DMCTO algorithm, DCORA demonstrates superior performance in optimizing task processing delay. The reason for this phenomenon is that DCORA can achieve perfect matching between task volume and computing resources. However, when using DMCTO for task computing, there will be mismatch between task volume and computing resources, resulting in an increase in calculation delay. Since the total amount of computational tasks is the same, the energy consumption of DMCTO and DCORA is basically the same.

Figure 5 represents the impact of data size on offloading cost. The change of data size will directly affect the change of delay and energy consumption. In order to highlight the impact of data size on offloading cost, we set the data volume for each computing task to be the same, with data sizes increasing from 1 Mbits to 4 Mbits. And the computing resource required for each task c_n is randomly distributed within the range set in Table 1. It can been seen that the DCORA scheme has the lowest offloading cost than other offloading schemes. This is because as the data volume increases, it allows UAV to choose appropriate strategies for data processing, effectively reducing task processing and energy consumption. With the increase in data volume, the advantages of the DCORA algorithm become more pronounced. Under high data volume scenarios, the feasibility and superiority of the DCORA algorithm can be demonstrated.

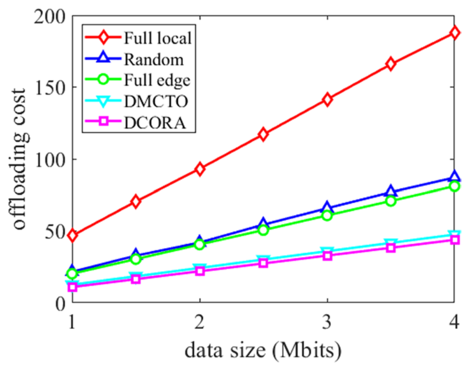

Fig. 5 Offloading cost under different data size and offloading mode

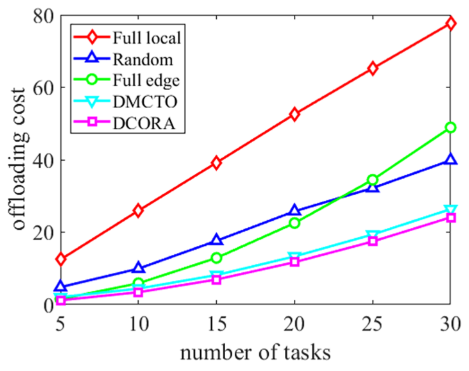

Fig. 6 Offloading cost under different number of tasks and offloading mode

Figure 6 reflects the impact of the number of offloading tasks on offloading cost. As the number of tasks to be offloaded increases, both computational and transmission pressures also increase. Therefore, we compared the global offloading costs under different offloading modes as the number of tasks varied from 5 to 30. It can be observed that the global offloading cost increases with the number of tasks increasing. And the DCORA algorithm having the smallest increase. The DCORA algorithm gradually pulls ahead as the number of tasks increases compared to other offloading schemes. This is because, in a fixed number of channels, an increase in the number of tasks leads to a decrease in transmission rates for devices. Simultaneously, the increasing number of computational tasks results in increased task processing delays and energy consumption. Additionally, when the number of tasks exceeds 25, all-edge offloading costs exceed the random decision mechanism. Due to the increase in the number of tasks, MEC computational resources cannot support all tasks. Random decision offloads some tasks to DCNs, alleviating the computational pressure on MEC. In summary, DCNs can effectively share the computational load of MEC, and the DCORA algorithm can minimize offloading costs by efficiently allocating communication and computational resources.

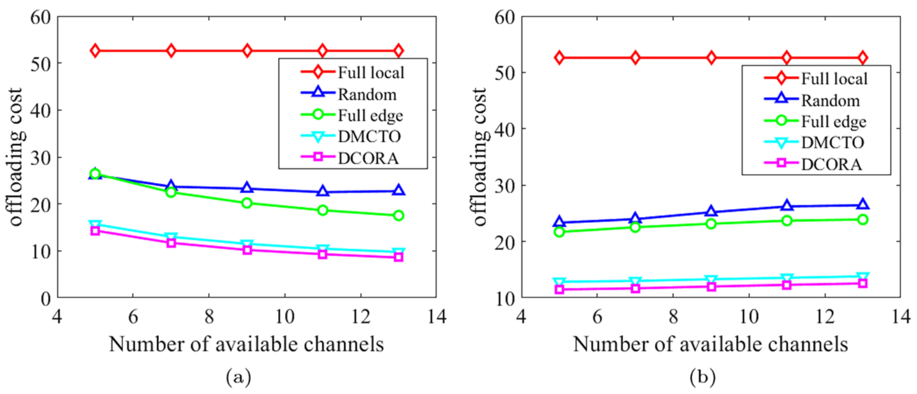

Fig. 7 **a** The offloading cost when sub-channels bandwidth is fixed and the number of channels varies, and **b** the offloading cost of changing the number of channels when the total channel bandwidth is fixed

Figure 7a and b depicts the impact of available channel count on offloading cost. The number of available channels and bandwidth is important component elements of communication resource. The change of channel resource will affect the transmission interference and then affect the transmission rate and delay. Figure 7a shows the variation trend of offloading cost under different numbers of channels, with the sub-channel bandwidth is fixed at 5 MHz. With the increase in the number of channels available to the devices, except for the local computing scheme, the offloading cost of other schemes shows a downward trend. This is because when selecting the offloading scheme, the option with the least co-channel interference was chosen, reducing task transmission delay and energy consumption. Figure 7b shows the variation trend with the total channel bandwidth fixed. The offloading cost slightly increases under various offloading mechanisms. Although the number of channels increases, prudent channel decisions lead to reduced co-channel interference. The DCORA algorithm consistently maintains the lowest offloading cost under varying channel conditions, demonstrating the effectiveness of the approach.

6 Conclusion

Aiming at the cloud-edge collaborative computing problem of air-ground integrated networks, we constructed an cloud-edge collaborative computing model. And we propose a computational offloading and resource allocation optimization scheme based on distributed game to minimize the offloading cost. Extensive simulation results show that the offloading cost increases when the number of tasks and the amount of data increase, and the changes of weight parameters and channel states affect the processing delay and energy consumption of the tasks. The DCORA scheme performs well in terms of convergence performance and convergence speed. The proposed scheme is able to significantly reduce the average task processing delay and always keep the minimum offloading cost compared the traditional schemes.

Acknowledgements

This work was supported in part by the Joint Fund of Equipment Pre-Research and Ministry of Education under Grant 8091B032131, and the Aeronautical Science Foundation of China under Grant 2020Z066050001.

Author Contributions

YP worked in supervision, project administration, resources and funding acquisition; XG helped in data curation, writing—review and editing and formal analysis; XZ helped in conceptualization, methodology and writing—original draft; LL helped in validation and formal analysis; CW worked in investigation and software and LH helped in resources and visualization.

Funding

This work was supported in part by the Joint Fund of Equipment Pre-Research and Ministry of Education under Grant 8091B032131, and the Aeronautical Science Foundation of China under Grant 2020Z066050001.

Availability of data and materials

Data will be made available on reasonable request.

Declarations

Competing interests

The authors declare that they have no competing interests.

Received: 20 October 2023 Accepted: 30 January 2024
Published online: 20 April 2024

References

1. Y. Wu, C. Cai, X. Bi, J. Xia, C. Gao, Y. Tang, S. Lai, Intelligent resource allocation scheme for cloud-edge-end framework aided multi-source data stream. EURASIP J. Adv. Signal Process. (2023). https://doi.org/10.1186/s13634-023-01018-x
2. Z. Su, Y. Wang, T.H. Luan, N. Zhang, F. Li, T. Chen, H. Cao, Secure and efficient federated learning for smart grid with edge-cloud collaboration. IEEE Trans. Ind. Inform. **18**(2), 1333–1344 (2022). https://doi.org/10.1109/TII.2021.3095506
3. Z. Zhou, Z. Jia, H. Liao, W. Lu, S. Mumtaz, M. Guizani, M. Tariq, Secure and latency-aware digital twin assisted resource scheduling for 5G edge computing-empowered distribution grids. IEEE Trans. Ind. Inform. **18**(7), 4933–4943 (2022). https://doi.org/10.1109/TII.2021.3137349
4. M. Jiménez-Guarneros, C. Morales-Perez, J.D.J. Rangel-Magdaleno, Diagnostic of combined mechanical and electrical faults in ASD-powered induction motor using MODWT and a lightweight 1-D CNN. IEEE Trans. Ind. Inform. **18**(7), 4688–4697 (2022). https://doi.org/10.1109/TII.2021.3120975
5. X. Liu, Q. Sun, W. Lu, C. Wu, H. Ding, Big-data-based intelligent spectrum sensing for heterogeneous spectrum communications in 5G. IEEE Wirel. Commun. **27**(5), 67–73 (2020). https://doi.org/10.1109/MWC.001.1900493
6. Z. Wang, H. Du, Q. Ye, HTR: a joint approach for task offloading and resource allocation in mobile edge computing, in *ICC 2021—IEEE International Conference on Communications* (2021), pp. 1–6. https://doi.org/10.1109/ICC42927.2021.9500595
7. M. Chen, S. Guo, K. Liu, X. Liao, B. Xiao, Robust computation offloading and resource scheduling in cloudlet-based mobile cloud computing. IEEE Trans. Mob. Comput. **20**(5), 2025–2040 (2021). https://doi.org/10.1109/TMC.2020.2973993
8. X. Liu, X.B. Zhai, W. Lu, C. Wu, QoS-guarantee resource allocation for multibeam satellite industrial internet of things with noma. IEEE Trans. Ind. Inf. **17**(3), 2052–2061 (2021). https://doi.org/10.1109/TII.2019.2951728
9. X. Liu, C. Sun, M. Zhou, C. Wu, B. Peng, P. Li, Reinforcement learning-based multislot double-threshold spectrum sensing with Bayesian fusion for industrial big spectrum data. IEEE Trans. Ind. Inf. **17**(5), 3391–3400 (2021). https://doi.org/10.1109/TII.2020.2987421
10. O. Karatalay, I. Psaromiligkos, B. Champagne, Energy-efficient resource allocation for D2D-assisted fog computing. IEEE Trans. Green Commun. Netw. **6**(4), 1990–2002 (2022). https://doi.org/10.1109/TGCN.2022.3190085
11. M. Chen, H. Wang, D. Han, X. Chu, Signaling-based incentive mechanism for D2D computation offloading. IEEE Internet Things J. **9**(6), 4639–4649 (2022). https://doi.org/10.1109/JIOT.2021.3107945
12. A.-E.M. Taha, N. Abu Ali, H.R. Chi, A. Radwan, MEC resource offloading for QoE-aware has video streaming, in *ICC 2021—IEEE International Conference on Communications* (2021), pp. 1–5. https://doi.org/10.1109/ICC42927.2021.9500696
13. W. Zhan, C. Luo, G. Min, C. Wang, Q. Zhu, H. Duan, Mobility-aware multi-user offloading optimization for mobile edge computing. IEEE Trans. Veh. Technol. **69**(3), 3341–3356 (2020). https://doi.org/10.1109/TVT.2020.2966500
14. L. Liu, J. Feng, X. Mu, Q. Pei, D. Lan, M. Xiao, Asynchronous deep reinforcement learning for collaborative task computing and on-demand resource allocation in vehicular edge computing. IEEE Trans. Intell. Transp. Syst. (2023). https://doi.org/10.1109/TITS.2023.3249745
15. L. Wang, G. Zhang, Deep reinforcement learning based joint partial computation offloading and resource allocation in mobility-aware MEC system. China Commun. **19**(8), 85–99 (2022). https://doi.org/10.23919/JCC.2022.08.007
16. X. Deng, J. Yin, P. Guan, N.N. Xiong, L. Zhang, S. Mumtaz, Intelligent delay-aware partial computing task offloading for multiuser industrial internet of things through edge computing. IEEE Internet Things J. **10**(4), 2954–2966 (2023). https://doi.org/10.1109/JIOT.2021.3123406
17. B. Zhang, L. Wang, Z. Han, Contracts for joint downlink and uplink traffic offloading with asymmetric information. IEEE J. Sel. Areas Commun. **38**(4), 723–735 (2020). https://doi.org/10.1109/JSAC.2020.2971807
18. W. Lu, X. Zhang, Computation offloading for partitionable applications in dense networks: An evolutionary game approach. IEEE Internet Things J. **9**(21), 20985–20996 (2022). https://doi.org/10.1109/JIOT.2022.3175729
19. X. Lv, H. Du, Q. Ye, TBTOA: A DAG-based task offloading scheme for mobile edge computing, in *ICC 2022—IEEE International Conference on Communications* (2022), pp. 4607–4612. https://doi.org/10.1109/ICC45855.2022.9838987
20. I.M. Ali, K.M. Sallam, N. Moustafa, R. Chakraborty, M. Ryan, K.-K.R. Choo, An automated task scheduling model using non-dominated sorting genetic algorithm II for fog-cloud systems. IEEE Trans. Cloud Comput. **10**(4), 2294–2308 (2020)
21. S. Dai, M. Li Wang, Z. Gao, L. Huang, X. Du, M. Guizani, An adaptive computation offloading mechanism for mobile health applications. IEEE Trans. Veh. Technol. **69**(1), 998–1007 (2020). https://doi.org/10.1109/TVT.2019.2954887
22. S. Dong, Y. Xia, J. Kamruzzaman, Quantum particle swarm optimization for task offloading in mobile edge computing. IEEE Trans. Ind. Inform. (2022). https://doi.org/10.1109/TII.2022.3225313
23. J. Yuan, Y. Xiang, Y. Deng, Y. Zhou, G. Min, Upoa: a user preference based latency and energy aware intelligent offloading approach for cloud-edge systems. IEEE Trans. Cloud Comput. (2022). https://doi.org/10.1109/TCC.2022.3193709
24. C. Yang, X. Chen, Y. Liu, W. Zhong, S. Xie, Efficient task offloading and resource allocation for edge computing-based smart grid networks, in *ICC 2019—2019 IEEE International Conference on Communications (ICC)* (2019), pp. 1–6. https://doi.org/10.1109/ICC.2019.8761535
25. Y. Liu, S. Xie, Q. Yang, Y. Zhang, Joint computation offloading and demand response management in mobile edge network with renewable energy sources. IEEE Trans. Veh. Technol. **69**(12), 15720–15730 (2020). https://doi.org/10.1109/TVT.2020.3033160
26. X. Gao, X. Huang, S. Bian, Z. Shao, Y. Yang, PORA: predictive offloading and resource allocation in dynamic fog computing systems. IEEE Internet Things J. **7**(1), 72–87 (2020). https://doi.org/10.1109/JIOT.2019.2945066
27. Y. Dai, K. Zhang, S. Maharjan, Y. Zhang, Deep reinforcement learning for stochastic computation offloading in digital twin networks. IEEE Trans. Ind. Inform. **17**(7), 4968–4977 (2021). https://doi.org/10.1109/TII.2020.3016320
28. W. Sun, H. Zhang, R. Wang, Y. Zhang, Reducing offloading latency for digital twin edge networks in 6G. IEEE Trans. Veh. Technol. **69**(10), 12240–12251 (2020). https://doi.org/10.1109/TVT.2020.3018817
29. T. Liu, L. Tang, W. Wang, Q. Chen, X. Zeng, Digital-twin-assisted task offloading based on edge collaboration in the digital twin edge network. IEEE Internet Things J. **9**(2), 1427–1444 (2022). https://doi.org/10.1109/JIOT.2021.3086961

30. G. Qu, H. Wu, R. Li, P. Jiao, DMRO: a deep meta reinforcement learning-based task offloading framework for edge-cloud computing. IEEE Trans. Netw. Serv. Manag. **18**(3), 3448–3459 (2021). https://doi.org/10.1109/TNSM.2021.3087258
31. M. Yu, A. Liu, N.N. Xiong, T. Wang, An intelligent game-based offloading scheme for maximizing benefits of IoT-edge-cloud ecosystems. IEEE Internet Things J. **9**(8), 5600–5616 (2020)
32. X. Xu, Q. Jiang, P. Zhang, X. Cao, M.R. Khosravi, L.T. Alex, L. Qi, W. Dou, Game theory for distributed IoV task offloading with fuzzy neural network in edge computing. IEEE Trans. Fuzzy Syst. **30**(11), 4593–4604 (2022)
33. P. Wang, N. Xu, W. Sun, G. Wang, Y. Zhang, Distributed incentives and digital twin for resource allocation in air-assisted internet of vehicles, in *2021 IEEE Wireless Communications and Networking Conference (WCNC)* (2021), pp. 1–6. https://doi.org/10.1109/WCNC49053.2021.9417521
34. X.-Q. Pham, T. Huynh-The, E.-N. Huh, D.-S. Kim, Partial computation offloading in parked vehicle-assisted multi-access edge computing: a game-theoretic approach. IEEE Trans. Veh. Technol. **71**(9), 10220–10225 (2022). https://doi.org/10.1109/TVT.2022.3182378
35. Q. Luo, C. Li, T.H. Luan, W. Shi, W. Wu, Self-learning based computation offloading for internet of vehicles: model and algorithm. IEEE Trans. Wirel. Commun. **20**(9), 5913–5925 (2021). https://doi.org/10.1109/TWC.2021.3071248
36. J. Huang, M. Wang, Y. Wu, Y. Chen, X. Shen, Distributed offloading in overlapping areas of mobile-edge computing for internet of things. IEEE Internet Things J. **9**(15), 13837–13847 (2022). https://doi.org/10.1109/JIOT.2022.3143539
37. P. Teymoori, A. Boukerche, Dynamic multi-user computation offloading for mobile edge computing using game theory and deep reinforcement learning, in *ICC 2022—IEEE International Conference on Communications* (2022), pp. 1930–1935. https://doi.org/10.1109/ICC45855.2022.9838691
38. R.N.K. Mensah, L. Zhiyuan, A.A. Okine, J.M. Adeke, A game-theoretic approach to computation offloading in software-defined D2D-enabled vehicular networks, in *2021 2nd Information Communication Technologies Conference (ICTC)* (2021), pp. 34–38. https://doi.org/10.1109/ICTC51749.2021.9441652
39. Y. Yang, C. Long, J. Wu, S. Peng, B. Li, D2D-enabled mobile-edge computation offloading for multiuser IoT network. IEEE Internet Things J. **8**(16), 12490–12504 (2021). https://doi.org/10.1109/JIOT.2021.3068722
40. W. Fan, L. Yao, J. Han, F. Wu, Y. Liu, Game-based multitype task offloading among mobile-edge-computing-enabled base stations. IEEE Internet Things J. **8**(24), 17691–17704 (2021). https://doi.org/10.1109/JIOT.2021.3082291
41. Q.-V. Pham, H.T. Nguyen, Z. Han, W.-J. Hwang, Coalitional games for computation offloading in NOMA-enabled multi-access edge computing. IEEE Trans. Veh. Technol. **69**(2), 1982–1993 (2020). https://doi.org/10.1109/TVT.2019.2956224
42. H. Ko, H. Lee, T. Kim, S. Pack, LPGA: location privacy-guaranteed offloading algorithm in cache-enabled edge clouds. IEEE Trans. Cloud Comput. **10**(4), 2729–2738 (2022). https://doi.org/10.1109/TCC.2020.3030817
43. X. Wu, B.S. Sharif, O.R. Hinton, An improved resource allocation scheme for plane cover multiple access using genetic algorithm. IEEE Trans. Evol. Comput. **9**(1), 74–81 (2005)
44. T. Fang, F. Yuan, L. Ao, J. Chen, Joint task offloading, D2D pairing, and resource allocation in device-enhanced MEC: a potential game approach. IEEE Internet Things J. **9**(5), 3226–3237 (2022). https://doi.org/10.1109/JIOT.2021.3097754
45. Y. He, J. Ren, G. Yu, Y. Cai, D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks. IEEE Trans. Wirel. Commun. **18**(3), 1750–1763 (2019). https://doi.org/10.1109/TWC.2019.2896999
46. F. Binucci, P. Banelli, P. Di Lorenzo, S. Barbarossa, Adaptive resource optimization for edge inference with goal-oriented communications. EURASIP J. Adv. Signal Process. (2022). https://doi.org/10.1186/s13634-022-00958-0