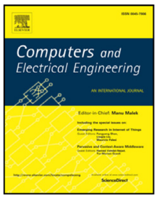
Computers and Electrical Engineering 87 (2020) 106764

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng



Computation offloading algorithm for cloud robot based on improved game theory

Fei Xu[a,b,*], Weixia Yang[a], He Li[a]

[a] Xi'an Technological University, Xi'an, China
[b] the State and Provincial Joint Engineering Lab. of Advanced Network, Monitoring and Control, China

ARTICLE INFO

Article history:
Received 20 November 2019
Revised 18 June 2020
Accepted 13 July 2020

Keywords:
Cloud robot
Edge cloud
Computation offloading
Game theory
Nash equilibrium

ABSTRACT

How to use the resources of the edge cloud more reasonably, reduce the energy consumption of machine equipment and ensure the shortest time for task completion are the challenges faced in cloud robot computation offloading research. In this paper, multiple heterogeneous cloud robot computing offloading problems are converted into game-type problems, and the computation-intensive tasks are divided to achieve partial offloading of tasks. An improved distributed game theory algorithm is designed to make each cloud robot's computation offloading strategy reaches the Nash equilibrium state, which maximizes the benefits of multiple participants, reduces the network load pressure of the central cloud, and reduces the transmission delay of computation offload. Simulation results show that the improved distributed game computation offload algorithm proposed enables cloud robots to reduce local computing energy consumption and shorten the average task completion time, greatly improving the edge cloud service quality.

1. Introduction

In recent years, the emergence of new applications such as face recognition, speech recognition, natural language processing, and virtual reality has caused the execution of related algorithms on embedded devices to consume a large amount of computing resources, while also meeting the user's requirements for low latency. Embedded devices are limited by their own computing, storage and other hardware resources when executing these new applications. Cloud robot [1] was proposed by Professor James Kuffner in 2010, which means that robots can offload computationally intensive tasks to the cloud, and use the rich computing resources of the central cloud to improve the quality of task completion. The traditional central cloud service model [2] has abundant computing resources, but it is easy to cause channel congestion and network delay. Edge computing [3] is an emerging computing model, which places computing, storage, bandwidth and other resources on the edge cloud near the device side to reduce transmission delay and bandwidth consumption. Cloud robots use computing offloading technology [4] to hand over some or all computing tasks to the cloud computing environment, which can help solve the problem of computing-intensive tasks for terminals with limited resources.

The cloud robot equipment divides computation-intensive tasks according to the offloading strategy, some tasks are left for local execution and some are uploaded to the edge cloud for cloud execution, as shown in Fig. 1. The edge cloud receives

* Corresponding author.
E-mail address: 1195862787@qq.com (F. Xu).

https://doi.org/10.1016/j.compeleceng.2020.106764
0045-7906/© 2020 Elsevier Ltd. All rights reserved.

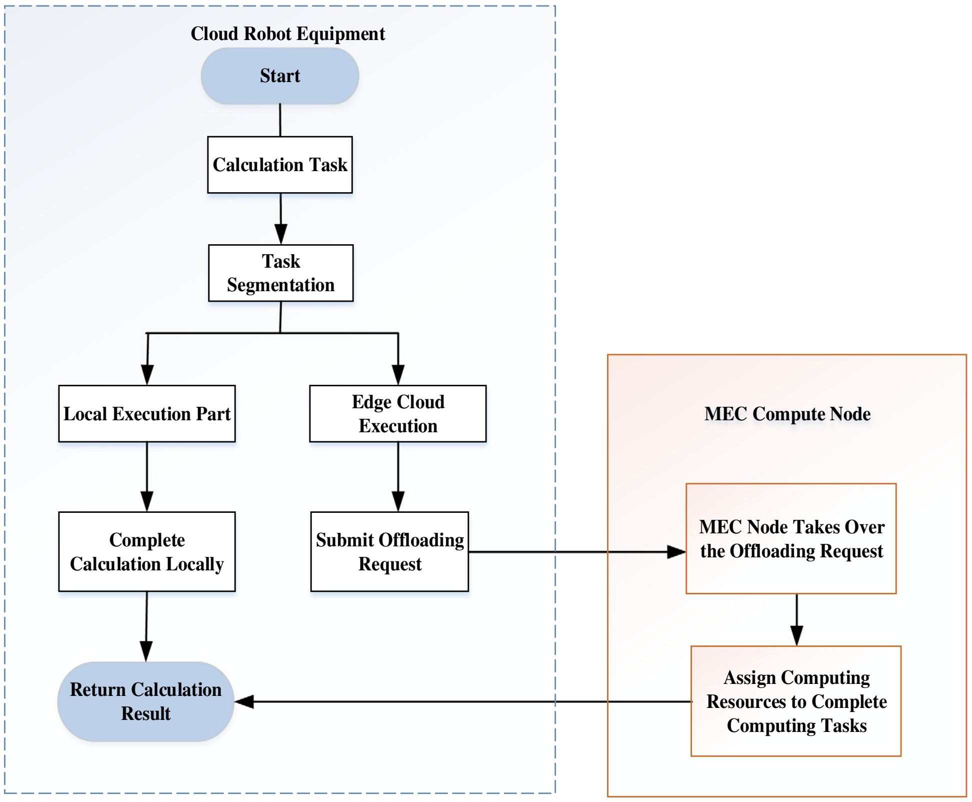

Cloud Robot Equipment
Start
Calculation Task
Task Segmentation
Local Execution Part
Edge Cloud Execution
Complete Calculation Locally
Submit Offloading Request
Return Calculation Result
MEC Compute Node
MEC Node Takes Over the Offloading Request
Assign Computing Resources to Complete Computing Tasks

Fig. 1. Calculation offloading process description.

the request from the cloud robot, executes it, and returns the result to the cloud robot. This can reduce the cloud robot's energy consumption and improve task execution efficiency.

The rest of the paper is organized as follows: section 2 analyzes the research status of computing offloading, section 3 establishes a mathematical model for calculating offloading, section 4 designs improved game theory algorithms, section 5 provides simulation experiments and discussion of experimental results, section 6 is a summary of the paper.

2. Related work

In computation offloading research, based on partial task offloading, You et al. [5] studied a multi-user MEC system and devised an optimal strategy to control the amount of offloaded data and time/sub-channel allocation to minimize system energy consumption. Tao [6] proposed a solution to the problem of minimizing energy consumption of users, and obtained an optimal offloading strategy by using the Lagrange multiplier method. This strategy determined the optimal amount of offloaded data and optimal allocation of computing resources. Examining the correlation between the intensive deployment of MEC servers and user computing tasks, Dai [7] proposed a two-tier computation offloading framework to solve the problem of load balance between multiple MEC servers and minimize system energy consumption by jointly optimizing user association, computing resource allocation, transmission power allocation, and computation offloading tasks. Jinlin Guo et al [8] proposed a heuristic collaborative computation offloading algorithm (HCCOA) for high real-time tasks, based on centralized cloud and edge cloud collaborative computing offloading with a goal to minimize the total processing time of tasks. This algorithm compares offloading strategies and selects the one with the shortest time. A genetic algorithm for collaborative computation offloading (GA4CCO) was also proposed to obtain a better task offloading strategy. Hongzhi Guo [9] et al. used the general structure of a hybrid fiber-wireless network, studied the problem of coordinated computing offloading based on MEC and CCC on this network structure, and defined it as a constrained optimization problem to minimize the total energy consumption of mobile devices. The approximation collaborative computation offloading (ACCO) scheme and distributed computation offloading algorithm (DCOA) were proposed.

In recent years, research on computation offloading algorithms based on game theory [10] has received more attention. Game theory refers to the use of strategies in games between multiple entities or teams under certain conditions. Competition or confrontational behavior becomes game behavior. The participants have different goals or interests. To achieve them, all of the parties must consider the various possible action plans of an opponent and try to choose the plan that is most beneficial or reasonable. Game theory studies whether a most reasonable behavior scheme exists for the parties to

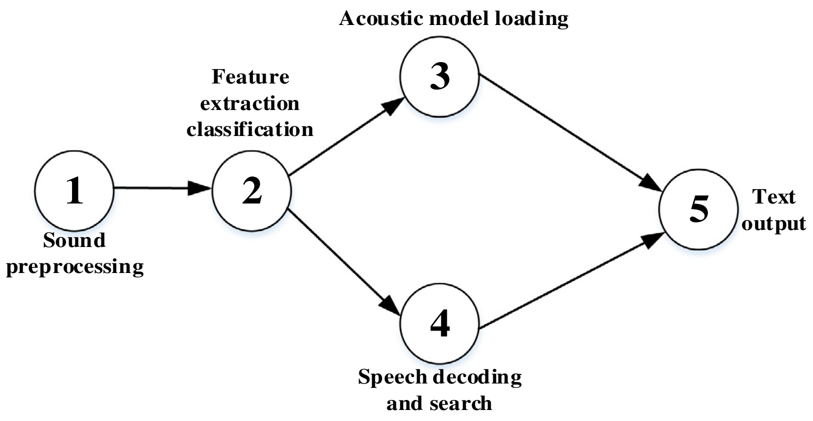


Fig. 2. Process of splitting up a task.

the struggle in game behavior, and seeks the mathematical theory and method of such a scheme. Chen [11] discussed the multi-user computation offloading problem of multi-access edge computing (MEC) in a multi-wireless channel environment, modeled the multi-user distributed computation offloading decision-making process as a multi-user computation offloading game, and designed a distributed computation offloading algorithm. Liu et al. [12] constructed a game model between mobile users and edge clouds, and provided their respective optimization goals and strategies. However, the focus is on the benefits of edge clouds to both gamers under unified pricing and differential pricing strategies. The influence of the game's Nash equilibrium strategy is not given. Kim et al. [13] used a Stackelberg game to model edge computing, and used the edge cloud and mobile users as the game leader and followers, respectively, to form a single-master and multi-slave game model. The problem is that the user's utility function does not consider task calculation delays, but only energy efficiency, which is not optimal for delay-sensitive applications. To improve task response time and resource utilization, Yingmo et al. [14] established a repeatable Stackelberg game-based edge computing task scheduling model between mobile users and edge servers, and adopted a historical dataset. The demand forecasting method solves the Stackelberg equilibrium for task scheduling. However, the task uninstallation is binary, and a task is indivisible for partial uninstallation. Xu et al. [15] established a Stackelberg equilibrium from the perspective of storage resource and bandwidth resource allocation of a mobile edge cloud. They did not consider the offloading decision of mobile user tasks or energy efficiency in the construction of utility functions. Aujla et al. [16] designed a Stackelberg game model with multiple dominant and multiple followers from the perspective of software-defined networks, discussed job scheduling and offloading schemes in a multi-edge cloud, and studied the impact of network performance on energy consumption and delay.

The research discussed above is primarily about partial or total computation offloading between a single-cloud server and multiple users, mainly considering cloud computing resource allocation [17] or wireless resources [18] optimization. However, little consideration has been given to the segmentation strategy, offloading time, offloading destination, and mapping relationship with multiple-edge cloud-server resources for the computation-intensive tasks of multiple heterogeneous users. This paper analyzes multiple heterogeneous machines in order to make full use of multiple-edge server resources, to discharge partial computationally intensive tasks, to uninstall the game design distributed computing algorithm for the offloading of Nash equilibrium and optimal strategy, and to minimize energy consumption of the robot and task completion time.

3. Computation offloading system model

Suppose there are $N = \{1, 2, 3, ..., n\}$cloud robot users, and each cloud robot runs a computationally intensive task. A task can be divided into $M = \{1, 2, 3, ..., m\}$ subtasks, which can be partially unloaded. Taking a class of industrial robot [19] as an example, the task consists of five subtasks: sound preprocessing, feature extraction and classification, acoustic model loading, speech decoding and searching, and text output. Each node represents a subtask. Under the condition of order constraints, a subtask cannot be executed before its predecessor subtask is completed, as shown in Fig 2.

The next section will analyze the communication method of the cloud robot and the edge cloud, the energy consumption and completion time of the cloud robot when performing tasks, and establish the system communication model, local computing model and edge computing model.

3.1. Communication model

The edge computing architecture is shown in Fig. 3. The cloud robot user interacts with the edge cloud through wireless communication. [20] Let $I = \{1, 2, ..., i\}$ represent multiple edge cloud servers, then $a_{m,n} = i$ indicates that device n offloads the computing task to edge cloud i, $a_{m,n} = 0$indicates that the task m of device n is executed locally, then the decision set

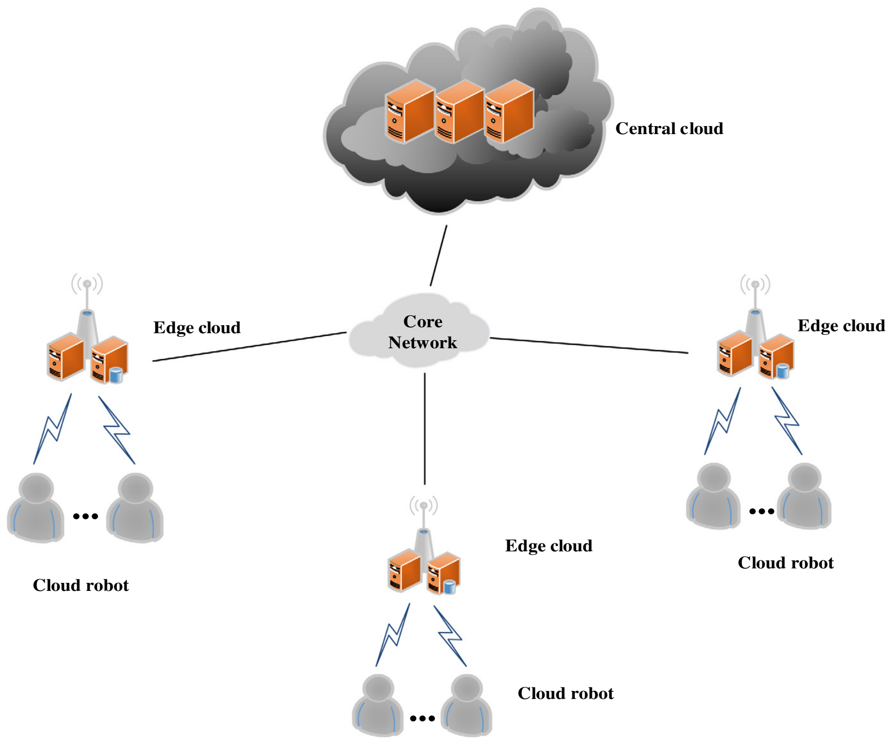


Fig. 3. Edge Computing Architecture.

for all tasks is represented by matrix A,

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}. \tag{1}$$

According to Shannon's law, [21] the data transmission rate when cloud robot users n is offloaded is:

$$\mathrm{R}_n^i(A) = w \log_2\left(1 + \frac{p_n^i Los_n^i}{\omega + \sum_{n=1}^{n} p_n^i Los_n^i}\right), \tag{2}$$

where:

(1) p_n^i is the transmission power when device n offloads the task to the edge cloud.
(2) Los_n^i is the channel gain, using wireless interference model based on cellular wireless environment. [22] We define $Los_n^i = d_n^{-\sigma}$ as a function of the distance of the user from edge cloud i, then d_n is the distance from device n to edge cloud i, and σ is 4.
(3) W is the channel bandwidth.
(4) ω is the Gaussian noise power inside the channel.

Known by (2) in $\frac{p_n^i Los_n^i}{\omega + \sum_{n=1}^{n} p_n^i Los_n^i}$, if multiple cloud robot users synchronize the calculation and offloading through the same edge cloud, the data transmission rate $\mathrm{R}_n^i(A)$ will decrease.

3.2. Local computing model

If a task is executed locally, then the cloud robot consumes energy during the calculation process, and the task completion time is the local calculation time. Suppose A calculates the amount of data to be offloaded for sub-task m of cloud robot n. Suppose $\mathrm{B}_{\mathrm{m,n}}$ is the amount of transmission data when computation sub-task m of cloud robot n is offloaded. $D_{m,n}$ is the number of CPU cycles required to complete the sub-task.

The local computing energy consumption is:

$$E_{m,n}^{loc} = V D_{m,n}, \tag{3}$$

where V is the energy consumption of each CPU cycle, and V is approximately linearly proportional to the square of the frequency. Then:

$$E^{loc}_{m,n} = \alpha (F^{loc}_{n})^{2} D_{m,n}. \tag{4}$$

When $\alpha = 2.5 \times 10^{-9}$, [23] $F^{loc}_{n}$is the computing power of cloud robot user n.
The local calculation time is:

$$T^{loc}_{m,n} = \frac{D_{m,n}}{F^{loc}_{n}}. \tag{5}$$

Definition 1: Wait time for sub-tasks. The wait time for a sub-task is the earliest time of completion for all direct precursor tasks. The wait time for sub-task m of local execution cloud robot user n is:

$$WT^{loc}_{m,n} = \max_{k \in pred(m)} \max \left\{ LT^{loc}_{m,n} \right\} \tag{6}$$

where $pred(m)$ is the direct precursor task set of task m.
Sub-task m of cloud robot n executes the completion time locally:

$$LT^{loc}_{m,n} = T^{loc}_{m,n} + WT^{loc}_{m,n}. \tag{7}$$

Assume the energy consumption and completion times represent the cost of local consumption:

$$C^{loc}_{m,n} = \lambda^{e} E^{loc}_{m,n} + \lambda^{t} LT^{loc}_{m,n}, \tag{8}$$

where λ^{e} and λ^{t} are the respective weighting factors of the energy consumption and delay.

3.3. Edge computing model

The cloud robot offloads a task to the edge cloud server for execution, and there is data consumption during the transmission process. The time to complete the task includes the data transmission time, edge cloud server computing time, and wait time, and the edge computation offloading model is established. The formulas are as follows.

(1) Energy consumption of cloud robot data transmission process:

$$E^{c}_{m,n} = p^{i}_{m,n} \times \frac{B_{m,n}}{R^{i}_{n}}. \tag{9}$$

(2) Data transmission time:

$$T^{tra}_{m,n} = \frac{B_{m,n}}{R^{i}_{n}}. \tag{10}$$

(3) Time of edge server task execution:

$$T^{c}_{m,n} = \frac{D_{m,n}}{F^{c}_{n}}. \tag{11}$$

(4) Time spent offloading a task in the cloud:

$$CT^{c}_{m,n} = T^{tra}_{m,n} + T^{c}_{m,n}. \tag{12}$$

(5) There are still wait times for the sub-tasks of the subsequent cloud robot n to be offloaded to the cloud:

$$WT^{c}_{m,n} = \max \left\{ T^{tra}_{m,n}, \max_{k \in reap(m)} CT^{c}_{k,n} \right\}, \tag{13}$$

so,

$$CT^{c}_{m,n} = T^{c}_{m,n} + WT^{c}_{m,n}. \tag{14}$$

The edge computing cost of the cloud robot in offloading is

$$C^{c}_{m,n} = \lambda^{e} E^{c}_{m,n} + \lambda^{t} CT^{c}_{m,n}. \tag{15}$$

This paper analyzes the energy consumption and task completion time when tasks are offloaded to edge cloud servers and when cloud robot machines are executed locally. We assign corresponding weighting factors for energy consumption and completion time and do normalization to represent the cost of the equipment, which is expressed by the function $C_{m,n}$.

4. The improved game model

Suppose $N = \{1, 2, 3, ..., n\}$ is a collection of game participants,

$Q = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$ is the strategic space of the game, and $I = \{0, 1, 2, ..., I\}$ is the game decision set.

The cost function is:

$$C_n = \lambda^e \sum_{m=1}^{m} E_{m,n} + \lambda^t \sum_{m=1}^{m} T_{m,n}, \tag{16}$$

where $0 \le \lambda^e \le 1, 0 \le \lambda^t \le 1, \lambda^e + \lambda^t = 1$.

To meet the needs of the cloud robot user, cloud robot users are allowed to select different weighting factors. For example, when the battery is low, a higher value of λ^e is assigned to save cloud robot user power. When the task of a cloud robot user is delay-sensitive, a higher value of λ^t is assigned to reduce the delay.

Our goal is to choose the appropriate strategy to minimize the cost of offloading. Hence the objective function is:

$$C_n(a_n) = \min\left\{C_n^{loc}, C_n^c\right\}. \tag{17}$$

Definition 2 Nash equilibrium [10]"

Strategy $Q^* = \begin{bmatrix} a_{11}^* & a_{12}^* & \cdots & a_{1n}^* \\ a_{21}^* & a_{22}^* & \cdots & a_{2n}^* \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1}^* & a_{m2}^* & \cdots & a_{mn}^* \end{bmatrix}$ is the Nash equilibrium of the distributed computation offloading game. If there are any offload policies:

$$C_n(a_{1n}^*, a_{2n}, ..., a_{mn}) \le C_n(a_{1n}, a_{2n}, ..., a_{mn}) \, \forall a_{mn} \in Q. \tag{18}$$

Once the calculation offload game reaches the Nash equilibrium, no user can deviate from the current calculation offload strategy to reduce costs.

If the Nash equilibrium is reached, then the following conditions will be satisfied:

Condition 1: When $a_{m,n} = 0$, in Nash equilibrium, for any $a_{m,n} \in \{1, 2, ...I\}$, there exists:

$$C_n^{loc} \le C_n^c. \tag{19}$$

Condition 2: When $a_{m,n} = i$, in Nash equilibrium, for any $a_{m,n} \in \{0, 1, 2, ..., i-1, i+1...I\}$, there exists $C_n^{loc} \ge C_n^c \ge C_n^i (c \ne i)$

Based on the limited improvement of the computational offload game, an updated algorithm of the computation offloading decision for heterogeneous mobile users is designed, including the optimization model and algorithm flow below.

4.1. Optimization model

Definition 3: Potential game: A potential game has a potential function $\psi(a)$ such that for arbitrary $n \in N$, $a_n^*, a_n \in Q_n$, if $C_n(a^*{}_n, a_{-n}) < C_n(a_n, a_{-n})$, then $\psi(a_n^*, a_{-n}) < \psi(a_n, a_{-n})$.

Definition 4: Finite exact potential function: A strategy a_n changes to a_n^* for a better response update if and only if the player's cost function decreases. In other words, if $C_n(a^*{}_n, a_{-n}) < C_n(a_n, a_{-n})$, then the finite precision potential function is represented as:

$$\begin{aligned} \psi(a_n) = \sum_{i=1}^{I} \left\{ \lambda^e \sum_{m=1}^{m} f(a_{m,n} = i) E_{m,n}^c + \lambda^t \sum_{m=1}^{m} f(a_{m,n} = i) LT_{m,n}^c \right\} \\ + \lambda^e \sum_{m=1}^{m} f(a_{m,n} = 0) E_{m,n}^{loc} + \lambda^t \sum_{m=1}^{m} f(a_{m,n} = 0) CT_{m,n}^c. \end{aligned} \tag{20}$$

If $a_{m,n} = =i$, then $f(a_{m,n} = i) = 1$, and otherwise, $f(a_{m,n} = i) = 0$.

The update strategy is:

$$\Delta a_n = \psi(a_n) - \psi(a_n'). \tag{21}$$

A greater value of Δa_n implies a greater chance of a cloud robot user update. Gamers with the largest difference will ultimately choose the appropriate offload strategy, and the computational offload game will reach Nash equilibrium after limited update iterations.

4.2. Algorithm design

```
Input: $N$, $B_{m,n}$, $D_{m,n}$, $F_n^{loc}$, $F_n^c$, $W$;
Output: $Q$, $C_n$;
Initialization: each cloud robot's tasks are executed locally, $a_{m,n} = 0$;
for each iteration t do;
{
        for each cloud robot n, $n \in N$;
{
        Collect current state calculations for offloading, and number of users per edge cloud load;
        Calculate the cost of cloud robots according to formulas (8) and (15);
        //Calculate the costs of local execution and cloud server execution;
        Choose a low-cost strategy $C_n(a_n) = \min\{C_n^{loc}, C_n^c\}$;
}
    If better update set $\zeta^t \neq \phi$ then:
{
        each cloud robot that satisfies $\zeta^t \neq \phi$ contends for the update opportunity;
        Calculation formula (20);//Calculate the cost of the cloud robot to complete the task;
        Comparison $\Delta a_n$, selecting the cloud robot user with the highest $\Delta a_n$ value. Update the policy $a_{m,n}$; //Different strategies have different
  costs, compare the difference, update the policy set
}
        Else
    Break;
}
        Of them, update set $\zeta^{t^\Delta} = \{a_{m,n}^* : C_{m,n}(a_{m,n}^*, a_{-m,n}) < C_{m,n}(a_{m,n}, a_{-m,n})\}$.
```

5. Simulation experiment

We used OpenStack, an open source cloud computing management platform, to build the edge cloud, and select the Medium virtual machine parameters in OpenStack to configure the parameters of the edge cloud. We set the kernel of the virtual machine to 2, with 12 GB of memory, a 30-GB hard drive, and a 5-GHz processor. Arduino is an open source prototyping platform that includes various versions of development boards and IDE kits. We used Arduino to create a cloud robot computing offload scenario. The experiment included 50 heterogeneous cloud robots, each consisting of a highly retractable car kit, the miniQ desktop robot chassis, Romeo V2 integrated controller, GP2Y0A21 distance sensor, and ESP8266 WiFi Be module. The computing power of each cloud robot was randomly selected from {0.5, 0.7, 1.0} GHz, and the locations were

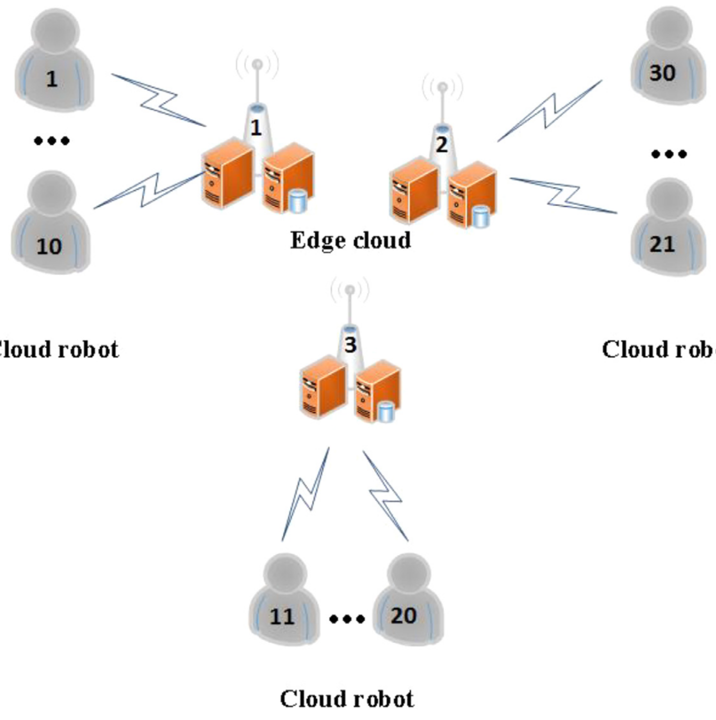


Fig. 4. Cloud robot computation offloading scenario.

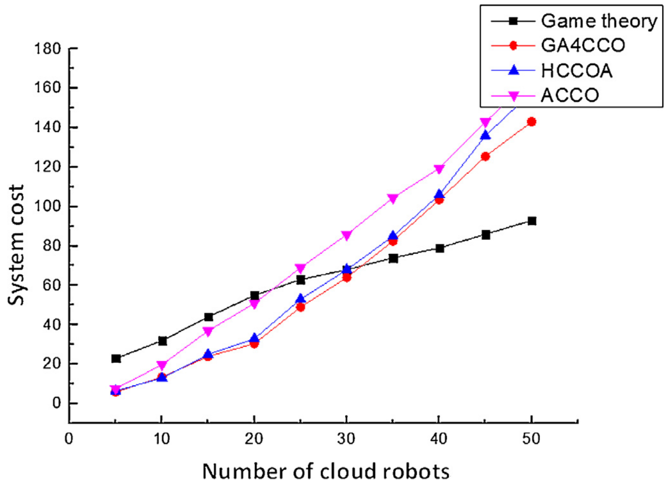


Fig. 5. System cost of different computation offloading algorithms.

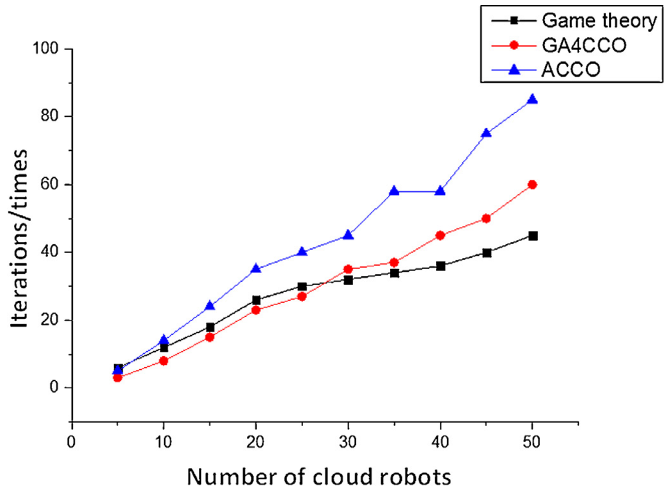


Fig. 6. Convergence trend of game theory algorithm.

randomly distributed in an area of 100 m by 100 m. The experiment then set three edge clouds in the center of the area, as shown in Fig. 4, regardless of the central cloud offloading problem.

Communication between a cloud robot and the edge cloud adopted a wireless access mode, with channel bandwidth $W = 5MHz$, transmission power $P_n = 100mW$, and channel internal Gaussian noise $\omega = -100dBm$. To evaluate the performance of the computation offloading strategy, the task of each cloud robot user was divided into five sub-tasks using a computational segmentation method [24]; the relationship between them is shown in Fig. 2. Each cloud robot sub-task data size was randomly distributed between 1 MB and 5 MB, and the required computing resources were set to 200 to 1,000 megacycles (the number of CPU cycles required to complete a task).

Simulation experiments were carried out to assess the system cost of a cloud robot, convergence of the game theory algorithm, probability of task offloading, and influence of the weighting factors.

We compared the improved game theory algorithm with HCCOA, [8] GA4CCO, [8] and ACCO. [9]· They are all recent studies that have achieved better results in reducing system energy consumption and delay in computing offloading. From Fig. 5, it can be seen that the improved game theory algorithm in this paper has the smallest average offloading cost. Compared with ACCO, it can reduce the offloading cost by 23% compared to ACCO, and by at least 3.8% compared to GA4CCO.

Next, we analyzed the convergence performance of the improved game algorithm. As shown in Fig. 6, the improved game theory algorithm had better convergence performance than GA4CCO and ACCO. In general, the improved game theory algorithm can improve the convergence performance by 8% compared to GA4CCO. Therefore, the improved game theory algorithm is more stable than GA4CCO and ACCO.

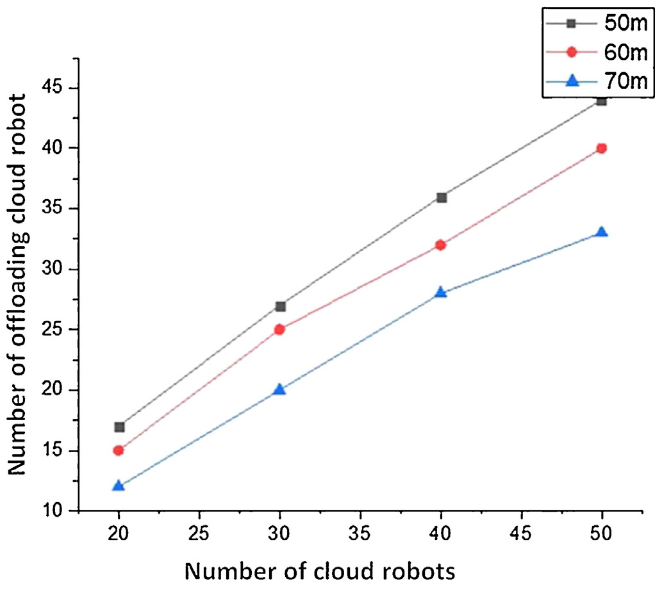


Fig. 7. Relationship between the number of offloading cloud robots and distance.

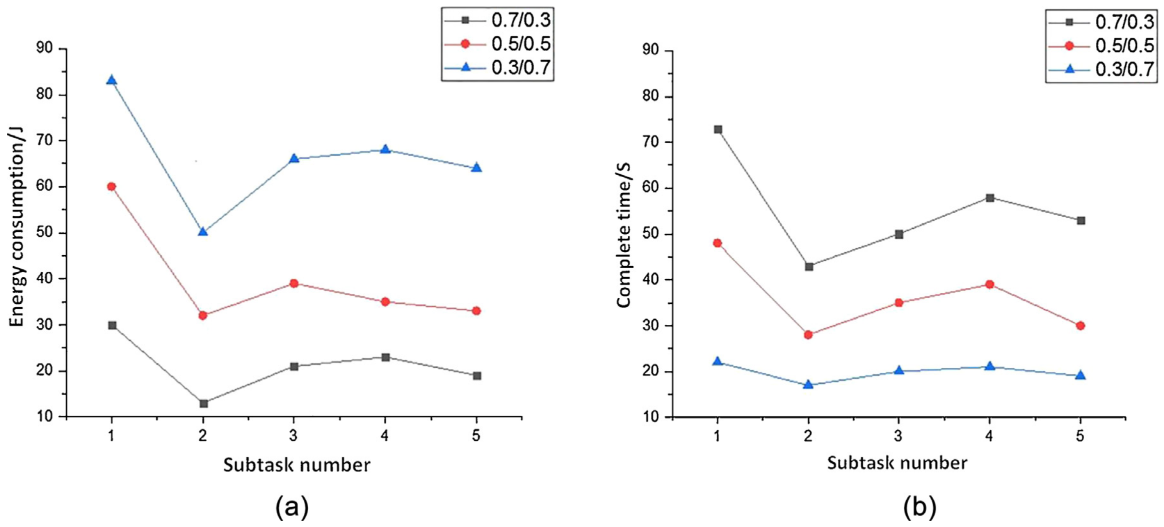


Fig. 8. (a). Relationship between weight factor and energy consumption.
Fig. 8 (b). Relationship between weight factors and task completion time.

The offloading efficiency of the algorithm was compared from the aspect of distance. The cloud robot adopts the principle of proximity, so that the closer the cloud robot user is to the edge cloud, the greater the probability of task offloading. As shown in Fig. 7, the closer the distance, the larger the number of cloud robots that can be unloaded, up to more than 90%.

Finally, the load input data rate DB is used to describe the complexity of the task, taking the five sub-tasks as the observation target. Define $DB_{m,n} = \frac{D_{m,n}}{B_{m,n}}$ for DB with values [11.056, 5.499, 4.742, 5.052, 3.676]. Compare the different weights of λ^e and λ^t, and complete the energy consumption and completion time of the task. As shown in Figs. 8(a) and (b), for a given task, energy consumption decreases and completion time increases as the energy factor λ^e increases. An increased λ^e will increase the probability that the task will be offloaded to the cloud, and the completion time of the task will increase when offloaded to the cloud. Therefore, λ^e and λ^t can be determined according to the actual situation.

This paper conducts numerical simulation experiments on the Cloudsim simulation platform. Through simulation experiments and data analysis, it can be seen that the offloading strategy of the improved game algorithm is less costly compared to the GA4CCO algorithm offloading strategy, and the game algorithm has a faster convergence speed and higher offloading efficiency. The improved game algorithm has the flexibility to select the optimal offloading strategy according to the state of each cloud robot and the situation of the edge cloud, to minimize energy consumption and task completion time.

6. Conclusion

In this paper, an improved game theory algorithm model is designed to assign different weight factors to cloud robots with different computing powers, so as to infer the best offload strategy and minimize the cost of computing offload.Simulation experiments analyze the cost of cloud robot systems with different calculation offload algorithms.The game theory algorithm proposed in this paper can reduce the system cost by 3.8% compared with the GA4CCO algorithm, and as the number of cloud robots increases, the advantages of the game theory algorithm in this paper are more obvious;In terms of algorithm convergence, the game theory algorithm can quickly reach a stable state, which is 8% faster than GA4CCO's convergence rate;In the scenario of multiple edge clouds, multiple cloud robots perform task offloading according to the principle of near-allocation. The closer the cloud robot is to the edge cloud, the greater the probability that the cloud robot can offload tasks. The highest offloading probability can reach 90%;Finally, it is verified that different weighting factors have an impact on energy consumption and completion time, and the energy consumption or delay can be reduced by adjusting the weighting factors.

The research work in this paper provides a basis for further exploration of cloud robot edge computing offloading.However, the impact of different partitioning methods on computing offload efficiency has not been considered; at the same time, the problem of edge cloud resource optimization scheduling and allocation has not been considered. These problems can be further explored in future research on computing offload.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Fei Xu: Conceptualization, Methodology, Validation. **Weixia Yang:** Formal analysis, Data curation. **He Li:** Software, Visualization.

Acknowledgement

This paper is funded by the following funds: Natural Science Project of Shaanxi Education Department (18JK0399) ; fund of the State and Provincial Joint Engineering Lab. of Advanced Network, Monitoring and Control, China (GSYSJ2018006).

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.compeleceng.2020.106764.

References

[1] Kuffner J J, Lavalle S M. Space-filling trees: A new perspective on incremental search for motion planning. IEEE/RSJ international conference on intelligent robots & systems; 2011.
[2] Zhou Yuezhi, Di Zhang. Near-end cloud computing: opportunities and challenges in the post-cloud computing era. Chin J Comput 2019;42(04):677–700.
[1] Shi Weisong, Zhang Xingzhou, Wang Yifan, Zhang Qingyang. Edge computing: current situation and prospect. Comput Res Dev 2019;56(01):69–89.
[2] You Changsheng, Zeng Yong, Zhang Rui. Resource management for asynchronous mobile-edge computation offloading. 2018 IEEE international conference on communications workshops (ICC workshops). IEEE; 2018.
[3] YOU C, HUANG K, CHAE H. Energy-efficient resource allocation for mobile-edge computation offloading. IEEE Trans Wirel Commun 2017;16(3):1397–411.
[4] TAO X, OTA K, DONG M. Performance guaranteed computation offloading for mobile-edge cloud computing. IEEE Wirel Commun Lett 2017;6(6):774–7.
[5] DAI Y, XU D, MAHARJAN S. Joint computation offloading and user association in multi-task mobile edge compu-ting. IEEE Trans Veh Technol 2018;67(12):12313–25.
[6] Guo Jinlin, Wu Jigang. Long collaborative computing offload algorithm based on optical fiber wireless network. Comput Eng Sci 2019;41(01):35–44.
[7] Guo Hongzhi, Jiajia Liu. Collaborative computation offloading for multi-access edge computing over fiber-wireless network. IEEE Trans Veh Technol 2018;67(5):1–13.
[8] XIN J, KWONG K Y, YONG Y.Competitive cloud resource procurements via cloud brokerage Proceeding of the 5th IEEE internationl conference on cloud computing technology and science. IEEE,2016:355-362.
[9] CHEN X, JIAO L, LI W. Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans Network 2016(5):2795–808.
[10] Liu M, Liu Y. Price-based distributed offloading for mobile-edge computing with computation capacity constraints. IEEE Wirel Commun Lett 2018;7(3):420–3.
[11] Kim S H, Park S, Chen M. An optimal pricing scheme for the energy efficient mobile edge computation offloading with OFDMA. IEEE Commun Lett 2018;21(9):1–14.
[12] Yingmo J, Xinyu T, Raymond K K. Online task scheduling for edge computing based on repeated stackelberg game. J Parallel Distrib Comput 2018;122(2):159–72.
[13] Xu X, Liu J, Tao X. Mobile Edge Computing enhanced adaptive bitrate video delivery with joint cache and radio resource allocation. IEEE Access 2017;5(99):16406–15.
[14] Aujla G S, Kumar N, Zomaya AY. Optimal decision making for big data processing at edge-cloud environment: an SDN perspective. IEEE Trans Ind Inform 2017;14(2):778–89.

[15] Lu H, Li Y, Mu S, Wang D, Kim H, Serikawa S. Motor anomaly detection for unmanned aerial vehicles using reinforcement learning. IEEE Internet Things J 2018;5(4):2315–22.
[16] Lu H, Li Y, Chen M, Kim H, Serikawal S. Brain intelligence: go beyond artificial intelligence. Mob Netw Appl 2018;23:368–75.
[17] Lu H, Wang D, Li Y, Li J, Li X, Kim H, Serikawa S, Humar I. CONet: A Cogn Ocean Netw 2019.
[18] Lu H, Li Y, Uemura T, Kim H, Serikawa S. Low illumination underwater light field images reconstruction using deep convolutional neural networks. Future Gen Comput Syst 2018;82:142–8.
[19] Rappaport T S. Wireless communications: principles and practice. Upper saddle river. NJ, USA: Prentice-Hall; 1996.
[20] Rappaport T. Wireless communications: principles and practice [M]. Electronic Industry Press; 2013.
[21] Miettinen AP, Nurminen JK. Energy effiffifficiency of mobile clients in cloud computing. In: Proceedings of the 2nd USENIX conference on hot topics in cloud computing. USENIX Association; 2010. 4-4.
[22] Lei Yang, Jiannong Cao, Hui Cheng. Multi-user computation partitioning for latency sensitive mobile cloud applications. IEEE Trans Comput 2015;64(8):2253–66.

Fei Xu, is an associate professor at Xi’an Technological University, a master’s tutor, and the director of the research and development center of the college. His research interests are edge computing, distributed systems, big data and adaptive middleware.

Weixia Yang, a master’s student at Xi’an Technological University, mainly studies cloud computing and edge computing.

He Li, a master’s student at Xi’an Technological University, mainly studies edge intelligence and deep learning.