Journal of Grid Computing (2023) 21:41
https://doi.org/10.1007/s10723-023-09673-y

RESEARCH

Data Security Aware and Effective Task Offloading Strategy in Mobile Edge Computing

Zhao Tong · Bilan Liu · Jing Mei · Jiake Wang · Xin Peng · Keqin Li

Received: 18 March 2023 / Accepted: 23 May 2023

Abstract With the research and development of 5G technology, emerging markets such as Wise Information Technology of med, smart transportation and industrial Internet are gradually growing, which not only provide convenience to people's life, but also put forward increasingly urgent demand for efficient parallel and distributed technologies. Therefore, in order to meet the need of high computing amount for application diversification, this paper proposes a novel scheduling solution with data security, aiming at simultaneously optimizing the system response time and the user's energy consumption. First, we model the scheduling problem in a mobile edge computing (MEC) environment as a Markov decision process (MDP) problem, and a three-tier collaboration model considering data security in the MEC environment is constructed. Second, the system response time and the energy consumption are simultaneously optimized in this paper, with objective weights which change in real-time. At the same time, load balancing at the edge layer is considered. Third, a deep reinforcement learning (DRL)-based secure offloading (DRLSO) algorithm is given as the solution for the research problem. In experiments from multiple angles, the proposed algorithm has good performance.

Keywords Collaborative optimization · Data security · Deep reinforcement learning (DRL) · Mobile edge computing (MEC) · Task offloading

Z. Tong (✉) · B. Liu · J. Mei · J. Wang · X. Peng · K. Li ·
College of Information Science and Engineering, Hunan Normal University, Changsha 410012, China

College of Information and Communication Engineering, Hunan Institute of Science and Technology, Yueyang 414006, China · Department of Computer Science, State University of New York New Paltz, New York 12561, USA

Z. Tong
e-mail: tongzhao@hunnu.edu.cn

B. Liu
e-mail: liubilan_0@hunnu.edu.cn

J. Mei
e-mail: jingmei1988@163.com

J. Wang
e-mail: 202170293854@hunnu.edu.cn

X. Peng
e-mail: peng-xin@foxmail.com

K. Li
e-mail: lik@newpaltz.edu

1 Introduction

The rapid development of information technology has brought people various application services, such as image recognition, audio processing, video processing, and augmented reality [8,38]. Correspondingly, higher requirements are also placed on the device's computing power. Even in this era, there have been significant breakthroughs in transistor technology, allowing CPUs to contain more transistors while maintaining the same package size [24]. The computing power of end devices

Published online: 10 July 2023

(EDs) has improved qualitatively and has specific processing capability; however, it still struggles to cope with the high computing demands brought by increasingly complex usage scenarios [21,33]. Therefore, this paper proposes a three-tier model for cooperation with a mobile edge computing (MEC) platform and a mobile cloud computing (MCC) platform with powerful computing capabilities. The device can upload tasks to platforms with excess computing capabilities. Through this collaborative mode, applications are no longer limited by resources and can complete multiple scenarios on the terminal, bringing richer experience to users.

The three-tier model combines the three layers' advantages to utilize resources fully. EDs can handle lightweight tasks without transmission with high security. MEC handles computationally intensive tasks, but it requires transmission. The computing power of MCC is more powerful than that of MEC, but the transmission is longer than MEC, and the transmission overhead is higher than the latter. Long-distance transmission means high transmission costs and higher security risks [5,29]. The attacker may intercept the data packet during the transmission of the task to obtain the data content [18]. For this reason, advanced encryption standard (AES) encryption technology is used in this paper to ensure the security of the task [1]. Therefore, if the task is offloaded to the remote, the processing costs are saved, but it generates additional overhead.

Based on the above situation, it is necessary to reasonably schedule limited computing resources [2,28]. A reasonable allocation strategy can find a better balance among contradictory goals, and has better performance on the premise of controlling costs. However, a poor strategy will weaken performance and waste resources [9,26]. Many methods are used to provide a solution, such as heuristic algorithms, numerical optimization methods, greedy algorithms. However, heuristic algorithm is proposed by experience, but lacks solid theoretical foundation. Numerical optimization method takes too long to solve problems. Greedy algorithms are often trapped in local optimal solutions and cannot obtain global optimal solutions. Therefore, this paper proposes an adaptive algorithm based on deep reinforcement learning to meet the needs of large-scale application scenarios, which can learn and adjust strategies adaptively. By sensing the current environment and combining the task's attributes, appropriate computing nodes are allocated to the task.

Thus, a deep reinforcement learning (DRL)-based secure offloading (DRLSO) algorithm is proposed in this paper to provide a solution for the research problem. The main contributions are as follows.

1. The heterogeneous three-tier task offloading model in MEC environment is constructed. This model considers data transmission mode when tasks are offloaded to the remote.
2. The task offloading problem in MEC environment is modeled as a Markov decision process (MDP), and an adaptive solution for simultaneously optimizing dual objectives is proposed.
3. The experiments are set up to verify the performance of the DRLSO algorithm, and the proposed algorithm shows a better performance than other comparison algorithms.

The rest of this paper is organized as follows. Section 2 discusses the current research status of this field. Section 3 shows the constructed three-tier scheduling model in the MEC environment and the objective function. Section 4 introduces the proposed algorithm and gives pseudocode. Section 5 designs experiments to test the performance of the proposed algorithm. Section 6 is a general summary of this paper.

2 Related Work

With the constant explosion of data volume, allocating appropriate computing nodes for tasks to ensure efficient utilization of limited resources has attracted widespread attention. Scholars have published many high-quality articles, and several recent works in related fields are listed below [4,10,27].

Yao *et al.* [34] considered a task assignment problem with path planning constraints among multiple robots. To solve the coupling problem, they proposed the homotopic method with two homotopy primitives, which performs better than traditional methods in experimental results. Sujaudeen *et al.* [25] constructed a task-aware resource allocation framework based on neural networks to allocate resources cost-effectively. The neural network classified tasks autonomously, and the proposed algorithm was applied to scheduling tasks, effectively reducing the probability of task migration. Chowdhary *et al.* [3] considered the problem of improving the quality of service (QoS) of educational services in the cloud Internet of Things environment. The tasks were prioritized at first. Then, a suitable machine were assigned to each task to speed up the execution of the tasks, achieving a throughput performance of more than 90 percent. Guo [6] proposed a scheduling algorithm based on fuzzy self-defense in cloud computing environment, which has a good optimization effect in terms of resource utilization rate, default rate and completion time. Zhang *et al.* [36] proposed a scheme that comprehensively considered task scheduling and containerization to improve the efficiency of edge servers. Simulation results show that the proposed scheme can effectively improve the execution efficiency and eliminate redundant container operations.

The research mentioned above proposed corresponding scheduling models based on problems in different application scenarios; however, it ignores the heterogeneity of computing resources. A single type of resource has its limitations. According to the characteristics of different resources, this paper proposes a model that combines the advantages of three different computing resources and cooperates to process tasks, so that the resources are used efficiently.

Xu *et al.* [32] proposed a reference vector-guided evolutionary algorithm based on the angle-penalty distance of normal distribution (RVEA-NDAPD) to prompt the multi-objective scheduling efficiency of the cloud server, and the proposed algorithm had a great performance compared to the MaOEAs method. Wang *et al.* [30] studied the joint optimization problem for task allocation and power allocation under different channel conditions and computing resources. The solution proposed based on the matching theory could effectively save the system's energy consumption, and have good adaptability to the system. Niu *et al.* [19] proposed a geo-aware workflow allocation method in a cloud environment, which reduced delay and traffic overhead of the system through reasonable task duplication. Liu *et al.* [15] introduced particle computing to divide tasks into three different types of tasks based on information particles. A scheduling strategy based on greedy strategy is proposed and its optimization effect on energy consumption is proved by numerical experiments. Hagras *et al.* [7] considered the problem of energy waste caused by replication in the repetitive list-based heuristic scheduling scheme, and proposes a mechanism to reduce repetition, which can effectively reduce energy consumption while maintaining delay.

The research mentioned above had designed an effective solution under the constructed scenarios. However, the solution proposed in this paper based on DRL has a stronger adaptive ability than traditional methods, and the application of neural networks has better performance even in the case of large data dimensions.

Shao *et al.* [23] considered the problem of optimizing the real-time performance of tasks in the online monitoring scenario in the smart grid, and built a scheduling model that considers cable distribution characteristics and task attributes in the MEC environment. The proposed method based on improved particle swarm optimization effectively reduced the average delay and improved the grid's reliability. From the perspective of a cloud service provider, Rekha *et al.* [22] studied the solution based on a genetic algorithm in the cloud environment to improve the performance of the cloud server while ensuring a fast response. Li *et al.* [14] built a multi-agent system (MAS) model and proposed a method to solve the multi-task multi-agent assignment problem, aiming at optimizing resource utilization, agent satisfaction, task assignment time, task completion time, etc. Moreover, the effectiveness of the proposed algorithm was verified in different scenarios. Li *et al.* [13] proposed a solution with high profitability and success rate in multi-agent systems, the proposed algorithm, which has a better stability system and reduces the total execution time. Kanemitsu *et al.* [11] considered the problem that shared containers, high parallelism and high resource utilization could not be satisfied simultaneously when scheduling containerized tasks in workflow. They proposed a cluster-based containerization scheduling scheme, which used shared containers to reduce the required computing power resources and thus reduce the task response time. Wang *et al.* [31] proposed a scheduling optimization model with caching mechanism to ensure privacy through decentralized offloading of tasks. The proposed solutions based on genetic differential evolution had a good performance in reducing latency.

The research mentioned above had good performance on the objective targets. However, in reality, there are a few cases where only one factor needs to be considered, and most need to balance contradictory goals. Therefore, the selection of weights indicating the importance of goals is also very important. The algorithm proposed in this paper calculates the weights through the objective weighting method, and can update the weights in real-time according to the status of the system.

3 The TCM Model

In this section, a three-tier model is introduced to describe the task scenarios. We first illustrate the components of the model in detail. Second, the entire workflow of the system is set forth through formal language. Third, the specific description of the problem studied in this paper, and the objective function are given.

3.1 System Model

The three-tier collaboration model in MEC (TCM) environment is considered in this paper. The three components of the TCM model are the local layer, the edge layer and the cloud layer. The TCM model is shown in Fig. 1.

- The local layer: This layer is composed of EDs in the area, including mobile phones, laptops, traffic lights, fax machines, etc. These devices are both generators of task requests and receivers of resulting data.
- The edge layer: This layer distributes edge servers, which are used to process task requests sent by EDs. In this paper, we design three edge regions with the same construction as an example to provide computing services for the upper layer.
- The cloud layer: This layer has quietly sufficient computing resources, but it needs to bear higher transmission costs. Therefore, it is suitable for processing computation-sensitive tasks but not transmission-sensitive tasks.

Those EDs at the local layer generate task requirements, denoted as $task = \left(id_{ed},\ id_t,\ ts_i,\ c_i,\ slot_i^{sub}\right)$, which represent the device that generates this task, the task identification, task size, the number of CPU revolutions required to calculate a unit of data, and the submission time, respectively. According to the characteristics of the task and the real-time status of the system, the task can be processed locally, on the edge, or on the cloud. When the task is selected to be offloaded to remote to process, the data will have an encryption process to ensure the security of the transmission process. Among them, tasks are divided into two types: cooperative tasks (CT) and non-cooperative tasks (NCT), which depend on the location of the ED which generates the task. If the ED is at the junction of multiple edge regions, we regard this task as a cooperative task, defined as

$$task_i = \begin{cases} CT & if\ loc_t^u \in A_j^{es} \cap A_k^{es},\ j \neq k, \\ NCT & other, \end{cases} \tag{1}$$

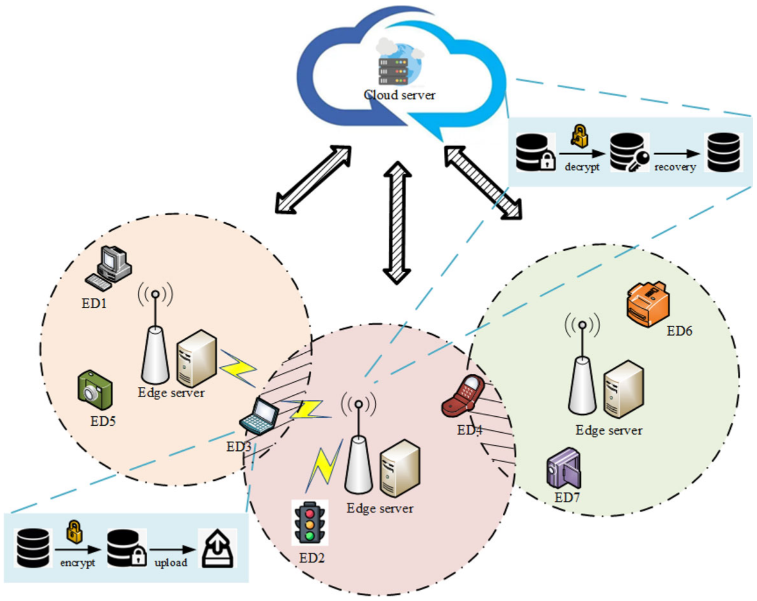

Fig. 1 The TCM scheduling model

where loc_t^u is the plane coordinates of the ED, and A_j^{es} and A_k^{es} are the service area of the given edge server. For cooperative tasks, on the basis of the original three options, they have additional multiple edge regions to choose from. The ED is in constant motion and its location is not fixed.

3.2 Response Time

Every once in a while the ED generates a task request. If the local computing power is sufficient and the task is processed locally, then no transfer is needed. In this case, the response time only includes local procession time, defined as

$$T_i^u = \frac{ts_i \times c_i}{f_u}, \tag{2}$$

where ts_i is the size of task i. c_i is the number of cpu cycles required to calculate one unit of data. f_u is the local computing frequency. Otherwise, the ED needs to encrypt the task data, and then send it to the remote to ensure the security of the transmission process.

It is a wireless transmission from the ED to the edge. The channel-to-interference plus-noise ratio (CINR) is defined as

$$\gamma_{e,b} = \frac{P_e g_{e,b}}{\sum_{i \in E, j \neq b} P_{\max} g_{i,j} + \sigma^2}, \tag{3}$$

where P_e is the ED transmit power. The channel gain $g_{e,b}$ is obtained through $g_{e,b} = \left(dist_{e,b}\right)^{-\delta}$, where δ is the path loss factor, and $dist_{e,b}$ is the distance between EDs and the edge. $\sum_{i \in E, j \neq b} P_{\max} g_{i,j}$ represents the inferences from EDs in other edge areas. σ^2 is the noise power. When multiple EDs in the same area access simultaneously, the data communication follows the connection mode of frequency division multiple access (FDMA) method. Therefore, the transmission in the region is independent of each other, and interference is not considered. However, interference from EDs in other regions at the same time is considered; Therefore, the wireless transmission ratio is

$$r_{e,b} = \frac{B}{U_b} \log_2\left(1 + \gamma_{e,b}\right), \tag{4}$$

where B is the bandwidth. U_b is the number of EDs in the same edge region at the same time.

Due to the limited capability of EDs, a lot of tasks need to be offloaded to the remote. However, frequent data interaction also brings data security risks. Therefore, secure transmission is meaningful to prevent the attacker from easily obtaining the data content when the user data is intercepted by malicious means. Tasks transmitted to the remote needs to be encrypted locally first. The encryption process is carried out on the network adapter, and the time consumed is

$$encr_i = \frac{ts_i}{nic_e}, \tag{5}$$

where nic_e is the rate of the local network adapter. After encryption, the size of the data may change, denoted as ts_i^{encr}. Until the task reaches the assigned processing device, it will be decrypted at the processor. The decryption time is

$$decr_i = \frac{ts_i^{encr}}{nic_j}, \tag{6}$$

where nic_j is the rate of the network adapter at the edge or the cloud.

If task is offloaded to the edge, the total communication time includes wireless transmission time, encryption and decryption time, defined as

$$comm_i^{e,b} = encr_i + \frac{ts_i^{encr}}{r_{e,b}} + decr_i, \tag{7}$$

$$= \frac{ts_i}{nic_e} + \frac{ts_i^{encr}\left(r_{e,b} + nic_j\right)}{r_{e,b} \times nic_j}, \tag{8}$$

where $\frac{ts_i^{encr}}{r_{e,b}}$ represents the transmission time between the base station (BS) and EDs. The total response time consists of two parts, the transmission part and the calculation part. Thus, the response time of the task when it is processed by the edge server is

$$T_i^o = comm_i^{e,b} + \frac{ts_i}{mach_j}, \tag{9}$$

where $mach_j$ is the processing rate of the assigned edge device, and $\frac{ts_i}{mach_j}$ is the calculation time.

However, if task is offloaded to the cloud, there is one more wire transmission process compared to offloading it to the edge. The channel from the BS to the cloud is a wired channel with time division multiplexed. Queuing may be required when tasks are intensive. The queuing time is

$$t_{queu} = \begin{cases} slot^{leis}_{fiber} - slot_{now} & if\ slot^{leis}_{fiber} \geq slot_{now}, \\ 0 & other, \end{cases} \tag{10}$$

where $slot^{leis}_{fiber}$ is the idle time of the channel, that is, the completion time of the last transmission task. $slot_{now}$ is the current time. The wired communication time is

$$comm_i^{b,c} = \frac{ts_i^{encr}}{r_{b,c}} + t_{queu}, \tag{11}$$

where $r_{b,c}$ is the rate of the wired channel. The total communication time for offloading to the cloud is

$$comm_i^{e,c} = encr_i + comm_i^{e,b} + comm_i^{b,c} + decr_i. \tag{12}$$

Therefore, the response time is

$$T_i^o = comm_i^{e,c} + \frac{ts_i}{mach_j}. \tag{13}$$

3.3 EDs Energy

In this paper, we only consider the energy consumption of the user side, mainly including local processing energy consumption and transmission energy consumption, and the energy consumption of the remote is not included in the calculation.

When the task is processed locally without offloading, only the local computing energy consumption is considered. The local power consumption is

$$P_u = \kappa\, (f_u)^3\,, \tag{14}$$

where κ is the effective switched capacitors in the chip. The energy consumption is

$$E_i^u = P_u \frac{ts_i \times c_i}{f_u}. \tag{15}$$

When the task is offloaded to the edge or cloud, only the encryption energy of the original data and the transmission energy of sending the task to the base station are calculated, defined as

$$E_i^o = P_u \times comm_i^{u,b}. \tag{16}$$

If tasks are offloaded to the cloud, the transmission from the BS to the cloud is done by the BS, so it does not count in this place.

3.4 Problem Description

We consider a scheduling scheme that simultaneously optimizes task response time and EDs energy in the TCM architecture. Task can be processed at local, the edge, or the cloud. In most cases, due to the limitation of the EDs hardware, tasks need to be offloaded to remote to process. However, when the remote provides sufficient computing power, it also bears the risk of communication costs and data security. Besides, the environment is changing dynamically. It is meaningful to synthesize the real-time status of all devices and task attributes to make strategies, so as to save the energy consumption of the client and shorten the response time as possible. In this process, some constraints are exist. The objective function and constraints can be expressed as

$$Min \sum_{i\in\mathbb{N}} \left(T_i^u + T_i^o\right), \sum_{i\in\mathbb{N}} \left(E_i^u + E_i^o\right), \tag{17}$$

$$s.t. \sum_{i\in\mathbb{N}} cpu_i \leq C, \tag{18}$$

$$\sum_{i\in\mathbb{N}} mem_i \leq M, \tag{19}$$

$$y \leq Y. \tag{20}$$

$\mathbb{N}$ is the collection of all tasks. cpu_i is the required CPU capacity, and C is the CPU capacity of the remote server itself. The mem_i is memory, and M is the server's inherent memory. y is the number of tasks to be processed in parallel, and Y is the maximum number of tasks allowed to be processed in parallel.

4 Proposed DRLSO Algorithm

In this section, the DRLSO algorithm is designed to solve the problem of task assignment for the purpose of reducing system delay and EDs energy to the greatest extent possible. First, the decision-making among edge layer is elaborated. Second, the whole workflow of DRLSO is spread out in detail.

4.1 Cooperation among Edge Areas

When the task is assigned to the edge layer for processing, it is still necessary to further determine the specific edge server region and specific computing node. The main steps are as follows.

- Obtain the current ED coordinates and the range of each edge service area, and determine which service areas the current ED is located in. According to Eq. (1), if there are multiple, then we regard it as a cooperative task; otherwise, we regard it as a non-cooperative task.
- For cooperative tasks, compute the load situation among each edge area to select a certain edge area to be offloaded to balance the load among regions, which can be expressed as

  $$ES_k = arg\min \sum_{j \in ES_k} \sum_{i \in \mathbb{N}} \frac{ts_i}{mach_j}, \qquad (21)$$

  where ES_k represents number of selected region.
- For non-cooperative tasks, directly select the edge area in which it is located.

The pseudocode of edge cooperation can be expressed as Algorithm 1.

Algorithm 1: The Edge Cooperative Algorithm

```
Input: Environment status
Output: Selected edge regions $k$
for Each $task_i$ do
    Get the $loc_t^u$ of the ED which generated $task_i$;
    for Each region $j$ do
        Judge the position relationship between $loc_t^u$ and $A_j^{es}$;
    end
    Judge the type of $task_i$ according to Eq. (1);
    if $task_i$ is cooperative then
        Choose the area $ES_k$ according to Eq. (21);
    else
        Ensure the area $k$ of the task;
end
```

4.2 DRLSO Algorithm

To better adapt to the requirements of reducing the response time and saving local energy consumption simultaneously, the DRLSO algorithm is developed in this paper. In the TCM environment, tasks can be processed locally, or offloaded to the edge or the cloud. Through the DRLSO algorithm, a sub-optimal strategy in the current environment can be quickly obtained to ensure that both objectives can achieve better results.

DRLSO is an algorithm based on DRL extension that adaptively learns and updates strategies by interacting with the environment. Reinforcement learning is a machine learning paradigm that learns through trial and error by constructing labels based on feedback from the environment. With each action taken, the environment is updated from the current state s to the next state s', as

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\left(Q_t^{tar}(s,a) - Q_t(s,a)\right), \qquad (22)$$

$$Q_t^{tar}(s,a) = R + \gamma \max_{a'} Q_t\left(s',a'\right), \qquad (23)$$

$$a = arg\max_{a'} Q_t\left(s',a'\right). \qquad (24)$$

$Q_t(s,a)$ is a value that represents the state of the environment at time t after taking action a in state s. $Q_t^{tar}(s,a)$ is the target state value of the environment. In this paper, the state is a value of the considered system response time and ED energy. a is a one-hot encoded one-dimensional array, like $a = [0,\cdots,1,\cdots,0]$. The length of the array is the number of optional computing nodes. $a[x] = 1$ indicates that the computing node with subscript x is selected. The environment evaluates the impact of the policy and gives the reward as

$$R = w_1 \times T_i + w_2 \times E_i, \qquad (25)$$

used as a feedback to the system, to better guide the decision making. w_1 and w_2 are the weight factor of each objective. However, large-scale task data can cause dimensional disasters for Q-table, which cannot be stored efficiently. Therefore, deep learning is introduced to replace Q-table with value function approximation. With the goal of maximizing R, the algorithm is able to continuously learn during the interaction to guide better decisions.

In the proposed algorithm, the secure transmission is considered using AES due to its features of security, high efficiency, easy implementation and flexibility [12,20]. A simple schematic diagram is shown in

Fig. 2. AES is a common symmetric encryption algorithm that is widely used in information protection, e-commerce, biometrics and other fields. At the sender, the plain text is encrypted according to the given key and the encryption function to obtain the cipher text. The key is the password used to encrypt the plain text, which is the same as the decryption key and cannot be easily leaked. Even if the encrypted cipher text is intercepted by an attacker, it cannot be cracked without the key. At the receiver, the cipher text is inversely transformed according to the key and the decryption function to obtain the original text.

The entire algorithm workflow is as follows. At the beginning, the ED side generates task requests. There may be multiple devices generating tasks at the same time, but the same device only generates one task at the same time. the real-time status of the system is obtained, and the evaluation tasks are processed locally, at the edge or in the cloud.

If the local computing resources are sufficient, the task is processed locally, and there is no need to encrypt the data for remote transmission. If the task is offloaded to the remote, it is necessary to obtain the wireless transmission rate according to the current system status, and the EDs encrypt the data before transmitting it to the BS. If task is offloaded to the cloud, compared to the edge, there will be a longer transmission distance from the BS to the cloud for wired transmission. If task is offloaded to the edge, comprehensively consider the location of the ED and the load of each edge server to choose the candidate edge region.

Based on the above information, the broker will output the assigned computing node a through Eq. (22).

When the task is processed, the result is returned to the ED. The change of environment state is perceived, and the feedback value is calculated by Eq. (25) to guide decision-making. Among them, w_1 and w_2 are the weight coefficients, calculated by the Entropy method [17] in real-time, which mainly divided into the following 4 steps.

- First, use min-max normalization to process the sample matrix, to remove the effect of dimensional differences on the results. All subsequent steps are based on the normalized value, defined as

$$r_i^k = \frac{r_i^k - \underset{i\in(1,n)}{Min}\left(r_i^k\right)}{\underset{i\in(1,n)}{Max}\left(r_i^k\right) - \underset{i\in(1,n)}{Min}\left(r_i^k\right)}, \tag{26}$$

  where n is the number of samples, and k represents the evaluation indicators.
- Second, for each indicator, calculate the weight of each sample corresponding to the indicator value, defined as

$$q_i^k = \frac{r_i^k}{\sum_{i=1}^{n} r_i^k}. \tag{27}$$

- Third, calculate the information entropy of each indicator, which represents the amount of information. The larger the entropy value is, the more valuable information it contains., defined as

$$en_k = -h\sum_{i=1}^{n} q_i^k \ln q_i^k, \tag{28}$$

  where h is $(\ln q)^{-1}$.
- Fourth, obtain the weight of the indicator according to the entropy value, defined as

$$w_k = \frac{1 - en_k}{n_k - \sum_{j=1}^{j=n_k} en_j}, \tag{29}$$

  where n_k is the number of indicators, and $n_k = 2$ in this paper.

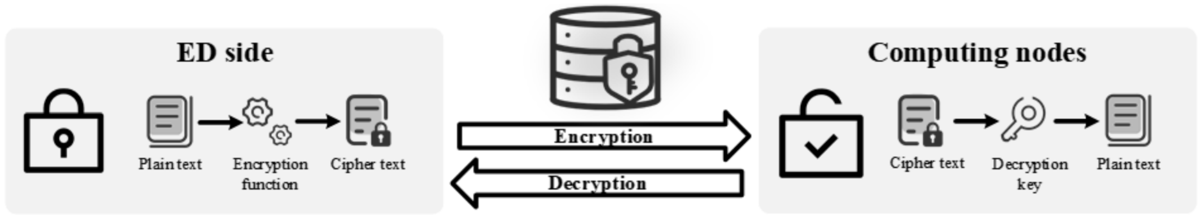

Fig. 2 The data encryption and decryption workflow of AES

Repeat the process above until the end. The pseudocode of the process is given in Algorithm 2.

Algorithm 2: The DRLSO Algorithm

```
Input:EDs
Output:Policy decisions
Initial the whole system environments;
Initialize the network structure;
for $task_i$ generated by EDs do
    Perceive the state of the environment $s$;
    Get candidate edge regions $ES_k$ through Algorithm 1;
    Get the selected computing node $mach_j$ through the DQN method ;
    if $mach_j$ is in the edge or the cloud then
        $task_i$ is offloaded to the remote;
        The ED side $u$ encrypts $task_i$;
        Calculate $r_{e,b}$ according to Eq. (3, 4) ;
        Calculate $T_i^o$, $E_i^o$ through Eq. (9, 13, 16);
    else
        $task_i$ is processed on the device $u$;
        Calculate $T_i^u$, $E_i^u$ through Eq. (2, 15);
    Calculate the feedback $R$ by environment according to Eq. (25);
    Update the network for optimization;
end
```

5 Experimental Verification and Performance Analysis

In this section, several experiments are conducted to prove the performance of the proposed DRLSO algorithm in the TCM environment. The configuration parameters of the computer are as follows: the operating system is Microsoft's win10 64-bit. The processor, CPU frequency, and running memory of the device are i5-8400 with six cores, 2.80 GHz, and 8 GB, respectively. First, the parameters related to the experiment are listed in Table 1. Second, the methods used in the algorithm are carefully selected through experiments. Third, we carefully observe and analyze the trend of the system under scene changes, and verify the performance of the proposed algorithm.

5.1 Activation Function

Neural networks are adopted to solve the problem of dimensional explosion when the amount of data is too large. However, the activation function plays a significant role in neural networks, which processes the signal output from the previous layer and transfers it to the next layer as the input. The signal processed in this way shows the non-linearity of the function, and the network can approximate any function. Due to the limitation of the linear function itself, only through this method can the neural network process more complex data. A good activation function can help the network fit better. To choose a suitable activation function, we design experiments to select from several common activation functions by observing the convergence of the network, including ReLu, Sigmoid, Leaky ReLu and Swish.

As shown in Fig. 3, the error value of the four methods has been reduced to a small value when the training iteration reaches hundreds of generations, reaching convergence. However, the error value oscillates in a small range after reaching convergence. The stability of the network is also a critical measure for the system. As seen from the partially enlarged image, when the

Table 1 Experimental settings

Symbol	Definition	Value
B	the bandwidth between ED and BS	10MHz[37]
N	the noise power of ED	-174dBm/Hz[16]
P_e	the transition power of ED	100mW[37]
c_i	the number of CPU cycles required to calculate a unit of data	500cycles/bit[37]
$r_{b,c}$	the up-link rate between BS and cloud	15Mbps
α	the learning rate	1e-4[35]
γ	the discount factor	0.7[35]
δ	the path loss factor	4[16]
κ	the effective switched capacitors in the chip	1e-28

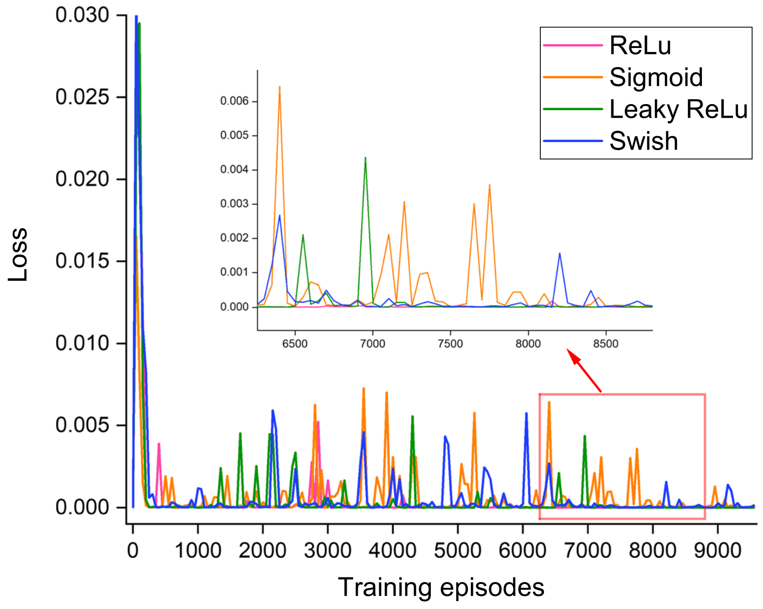

Fig. 3 Convergence of different activation functions

training process reaches a later stage, the fluctuation of ReLu is the smallest, followed by Leaky ReLu and Swish, and the worst stability is the Sigmoid function. This is due to the characteristics of the Sigmoid function itself, which is prone to the problem of gradient disappearance.

5.2 Weighting Method

In this paper, the weighted sum of the two objectives as the reward value is designed to measure the pros and cons of the current choice on the future impact. The weight coefficient indicates the degree of emphasis on the objectives, and plays a guiding role in the training of the system. Thus, it is crucial to dynamically adjust the two weight coefficients according to the real-time situation of the system. Considering this scenario, multiple objective weight determination methods are considered to guarantee the reasonableness of the weight value.

From Eq. (25), the reward value ranges from -1 to 0. The larger the reward value is, the more beneficial the broker thinks the current decision will be in the future. Figure 4 shows the guiding influence of six weight determination methods of entropy, variation coefficient, principal component analysis, critic weight, independence weight, and factor analysis on the future decision of the system. According to the feedback performance of the system, the experimental results of these six methods are ranked according to their performance. It can be seen that the reward value returned by the broker increases as the training progresses. When training to the 5000th round, the performance of each weight algorithm tends to be stable. Among them, the system responds best to the entropy method in Fig. 4(a). The entropy has the least jitters and remains at a high reward value, higher than -0.05. The worst is the performance of the factor method in Fig. 4(f), which has the highest frequency of oscillation. This is determined by the core idea of each weighting method. Therefore, the entropy method is adopted in this paper.

5.3 Edge Collaboration

The TCM model raised in this paper contains two parts of collaboration. One part is the collaboration of EDs, edge servers, and cloud servers. The other part is the collaboration of different edge servers due to the overlapping of multiple edge service areas. To balance the amount of calculation among regions as much as possible, the load balancing is considered when designing the DRLSO algorithm. The load calculation method for each region is

$$reg_k^{load} = \sum_{i \in T} \frac{ts_i}{mach_k^j}, \tag{30}$$

where T is the set of all tasks assigned to all machines in the kth region. The performance of the proposed algorithm is verified through experiments in this section.

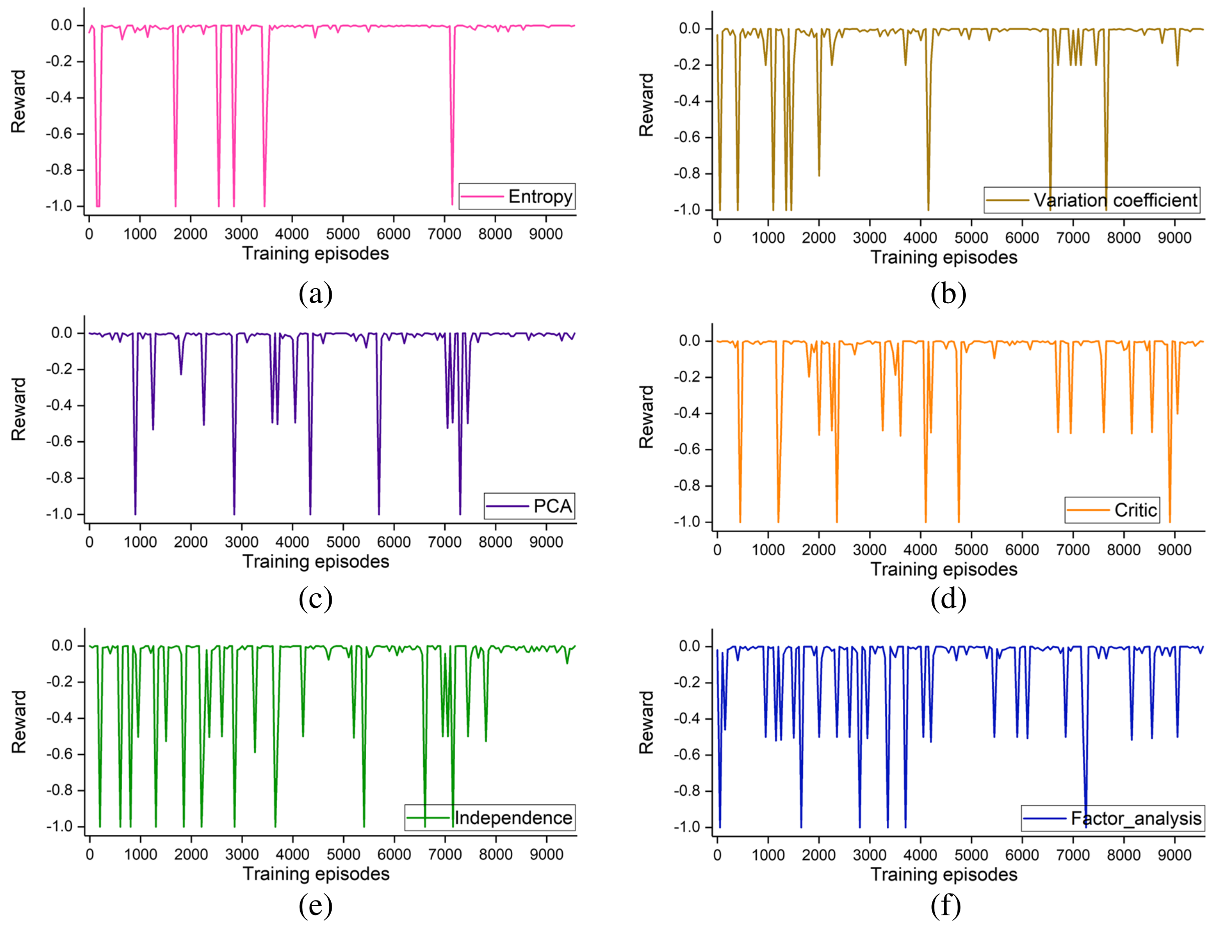

Fig. 4 The impact of different weight methods on the system. (a) Entropy. (b) Variation coefficient. (c) Principal components analysis. (d) Critic weight. (e) Independence weight. (f) Factor analysis

In Fig. 5, the experiment tests two different load situations with the expansion of the user scale. The experiment takes three edge regions as an example, and these three regions have overlapping parts and independent parts. The major premise is that these two conditions are all three-terminal synergy, and the only difference is whether the edge layer is synergistic or not. As the population of EDs increases, the load in each region also increases steadily, because the volume of tasks also expands. However, it is evident that in the condition of edge coordination, the situation of regional load imbalance becomes increasingly severe with the expansion of EDs. In contrast, the regional load considering edge coordination is much more balanced, and even when the task scale is enlarged, it also shows better stability.

5.4 Task Distribution

In the TCM environment, tasks can be processed locally, at the edge or in the cloud. This structure is designed to give full play to the computing resources of each layer, and avoid waste of resources or tension in only a certain part. Thus, in this section, we want to observe that when the task becomes increasingly intensive, the task is more inclined to be assigned to which side to the process.

The horizontal axis is the task generation rate, which is proportional to the number of tasks generated in the same time period, for the fixed number of devices in this experiment. From Fig. 6, it can be seen that more tasks are inclined to be offloaded to the edge,

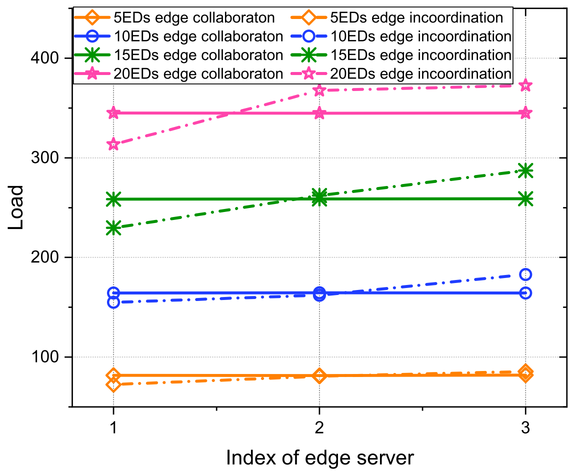

Fig. 5 Change with EDs number and service load under the influence of edge coordination

while the processing volume of the local and cloud has not changed much. This is because this paper aims to optimize the local-side experience, response time and local-side energy consumption. As the volume of tasks enlarges, the ED side can not afford additional tasks because of its limited capacity. Tasks are reluctant to move the cloud because the distance between EDs and the cloud is much farther than that from EDs to the edge side, prolonging task delay.

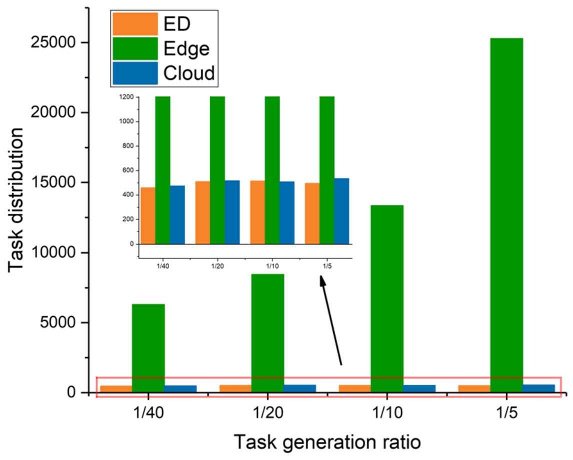

Fig. 6 Task distribution of three processing sides under the change of task density

5.5 Local Capability Variation

In fact, the CPU, hard disk, memory, and other hardware on the market are dazzling, as well as the network environment and other factors, resulting in different computing capabilities of terminal devices. In such a background, the purpose of this section is to observe what the impact is on a variety of EDs.

Figure 7 shows the impact of EDs capability changes on two objectives, the system response time and the energy consumption of EDs. From the graph above, it can be seen that the trends of energy and response time are opposite to each other. As the capacity of EDs increases, the response time decreases sharply in the early stages, and the growth in energy does not increase much. However, when the computing power of the ED exceeds a certain value in (0.2, 0.6), this phenomenon has changed. As can be seen from the slope of the image, the energy consumption increases faster than the slope decreases, which means a surplus of resources. Therefore, the larger capacity of ED is not the better, and an appropriate value can maximize the benefit.

5.6 Computing Node Expansion

In order to more comprehensively verify the performance of the proposed algorithm, in addition to the expansion of the scale of ED, the expansion of service nodes is also a situation that needs to be considered.

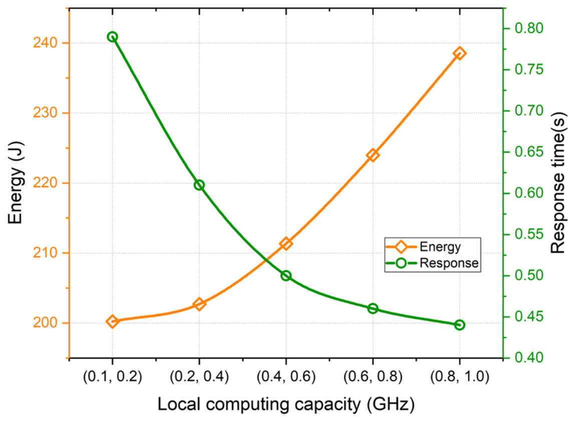

Fig. 7 Changes in response time and energy under miscellaneous local capacities

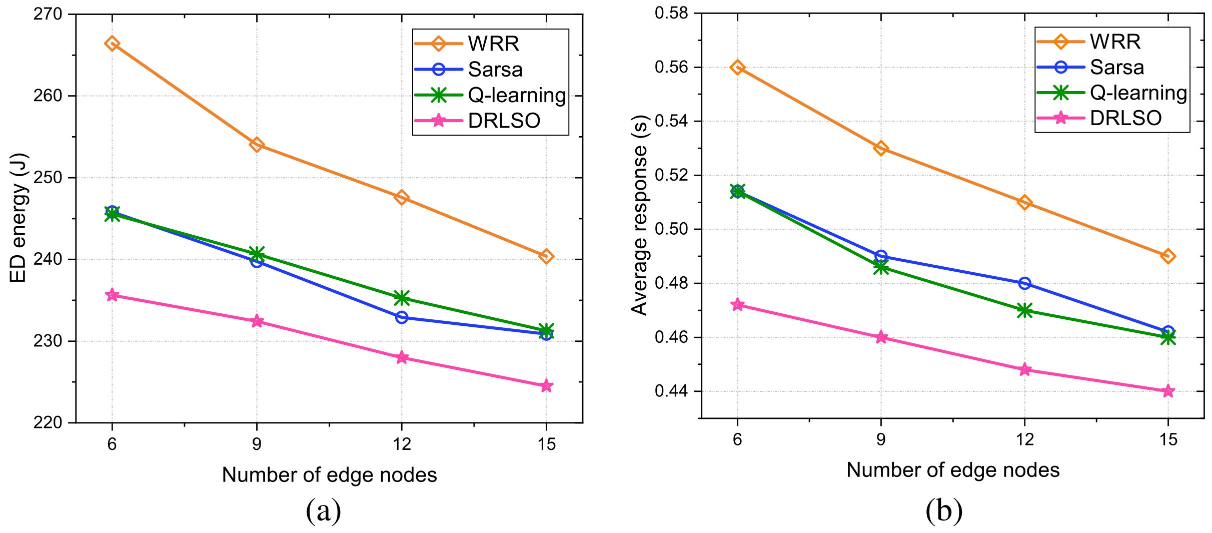

Fig. 8 Optimization effects of different algorithms on response time and energy under the change of number of computing nodes. (a) Optimization effect on energy. (b) Optimization effect on response time

Besides, in the experiment, three additional algorithms are selected to compare with the DRLSO algorithm in the same experimental environment.

Figure 8 shows the optimization effects of the four algorithms on local energy consumption and response time in the case of computing node expansion. Since the computing power of the cloud is sufficient in the setting of this paper, the computing power expansion of the edge layer is mainly considered in this experiment. The DRLSO algorithm achieves the best results in both local energy saving and latency reduction, with efficient reductions as the number of serving nodes increases. However, the weighted round robin (WRR) has the worst performance in saving user energy consumption and reducing latency, because it ignores real-time changing environmental conditions when assigning tasks. The effect of the Q-learning is basically similar to that of the Sarsa, and this is because both of them belong to reinforcement learning methods, lacking good adaptability when the task volume is large. Thus, the proposed DRLSO algorithm shows better performance for objective optimization and better robustness when scaled up.

6 Conslusion

In this paper, we construct a three-tier scheduling model in the MEC environment, and propose the DRLSO algorithm, which simultaneously saves terminal energy consumption and reduces system delay in allocating tasks in real-time. The TCM model combines three types of resources to process tasks collaboratively, giving full play to their respective advantages. For tasks that require cross-platform transmission for processing, the AES encryption method is performed on the data to ensure data security. As tasks are processed, the task offloading strategy is continuously learned from experience and adjusted in real-time. The weights of the two objectives are obtained in real-time through the objective weighting method, and the appropriate weighting method is selected through experimental verification to accelerate the convergence of the network and obtain higher reward values. Through simulations, the proposed DRLSO algorithm performs better than the comparison algorithm.

In many cases, there will be resource preemption between tasks due to the characteristics of different tasks. Therefore, in the future, we will focus on more complex and realistic task scenarios and study suitable scheduling schemes.

Acknowledgements The authors thank the editor and reviewers for their insightful comments and valuable suggestions

Author contributions Zhao Tong contributed to the conception of the study and wrote fifty percentage of the manuscript. Bilan Liu wrote fifty percentage of the manuscript, performed modeling design and construction, and also conducted part of the experiment. Jing Mei contributed significantly to the conception, analysis and algorithms design of the study. Jiake Wang performed the data analyses and English grammar checking. Xin Peng helped perform the analysis with constructive discussions. Keqin Li helped improve the English grammar and checked spelling

Funding Information This work was supported by the Program of National Natural Science Foundation of China (grant No. 62072174, 61502165), Outstanding Youth of Hunan Province (grant No. 2023JJ10030), National Natural Science Foundation of Hunan Province, China (grant No. 2022JJ40278, 2020JJ5370), Scientific Research Fund of Hunan Provincial Education Department, China (Grant No. 22A0026)

Declarations

Competing Interests The authors declare that they have no competing interests

References

1. Anoop, S., Singh, J.: Multi-user energy efficient secured framework with dynamic resource allocation policy for mobile edge network computing. J. Ambient Int. Humanized Comput. **12**(7), 7317–7332 (2021)
2. Chai, X.: Task scheduling based on swarm intelligence algorithms in high performance computing environment. J. Ambient Int. Humanized Comput. 1–9 (2020)
3. Chowdhary, S.K., Rao, A.L.N.: Qos enhancement in cloud-iot framework for educational institution with task allocation and scheduling with task-vm matching approach. Wirel. Personal Commun. **121**(1), 267–286 (2021)
4. Geng, X., Mao, Y., Xiong, M., Liu, Y.: An improved task scheduling algorithm for scientific workflow in cloud computing environment. Cluster Comput. **22**(3), 7539–7548 (2019)
5. Guo, K., Quek, T.Q.S.: On the asynchrony of computation offloading in multi-user mec systems. IEEE Trans. Commun. **68**(12), 7746–7761 (2020)
6. Guo, X.: Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm. Alexandria Eng. J. **60**(6), 5603–5609 (2021)
7. Hagras, T., Atef, A., Mahdy, Y.B.: Greening duplication-based dependent-tasks scheduling on heterogeneous large-scale computing platforms. J. Grid Comput. **19**(1), 13 (2021)
8. Haines, R., Scamell, R.W., Shah, J.R.: The impact of technology availability and structural guidance on group development in workgroups using computer-mediated communication. Inf. Syst. Manag. **35**(4), 348–368 (2018)
9. Han, M., Zhang, T., Lin, Y., Deng, Q.: Federated scheduling for typed dag tasks scheduling analysis on heterogeneous multi-cores. J. Syst. Architecture **112**, 101870 (2021)
10. Islam, N., Azim, A.: A situation-aware task model for adaptive real-time systems. J. Ambient Int. Humanized Comput. **11**(10), 4249–4259 (2020)
11. Kanemitsu, H,. Kanai, K., Katto, J., Nakazato, H.: A containerized task clustering for scheduling workflows to utilize processors and containers on clouds. J. Supercomput. pp 1–45 (2021)
12. Ke, Y., Xia, X.: Timed automaton-based quantitative feasibility analysis of symmetric cipher in embedded rtos: A case study of aes. Sec. Commun. Netw. 2022 (2022)
13. Li, M., Liu, C., Li, K., Liao, X., Li, K.: Multi-task allocation with an optimized quantum particle swarm method. Appl. Soft Comput. **96**, 106603 (2020)
14. Li, M., Wang, Z., Li, K., Liao, X., Hone, K., Liu, X.: Task allocation on layered multiagent systems: When evolutionary many-objective optimization meets deep q-learning. IEEE Trans. Evol. Comput. **25**(5), 842–855 (2021)
15. Liu,S., Ma, X., Jia, Y., Liu, Y.: An energy-saving task scheduling model via greedy strategy under cloud environment. Wirel. Commun. Mob. Comput. pp 1–13, (2022)
16. Mao, Y., Zhang, J., Song, S.H., Letaief, K.B.: Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems. IEEE Trans. Wirel. Commun. **16**(9), 5994–6009 (2017)
17. Naik, B.B., Singh, D., Samaddar, A.B.: Multi-objective virtual machine selection in cloud data centers using optimized scheduling. Wirel. Personal Commun. **116**(3), 2501–2524 (2021)
18. Nawrocki, P., Pajor, J., Sniezynski, B., Kolodziej, J.: Modeling adaptive security-aware task allocation in mobile cloud computing. Simul. Model. Pract. Theory. **116**, 102491 (2022)
19. Niu, M., Cheng, B., Feng, Y., Chen, J.: Gmta: a geo-aware multi-agent task allocation approach for scientific workflows in container-based cloud. IEEE Trans. Netw. Serv. Manag. **17**(3), 1568–1581 (2020)
20. Osvik, D.A., Shamir, A., Tromer, E.: Cache attacks and countermeasures: the case of aes. In Cryptographers' track at the RSA conference, pp 1–20 Springer, (2006)
21. Qiao, X., Ren, P., Dustdar, S., Liu, L., Ma, H., Chen, J.: Web ar: A promising future for mobile augmented reality-state of the art, challenges, and insights. Proceed. IEEE **107**(4), 651–666 (2019)
22. Rekha, P.M., Dakshayini, M.: Efficient task allocation approach using genetic algorithm for cloud environment. Cluster Comput. **22**(4), 1241–1251 (2019)
23. Shao, S., Zhang, Q., Guo, S., Qi, F.: Task allocation mechanism for cable real-time online monitoring business based on edge computing. IEEE Syst. J. **15**(1), 1344–1355 (2020)
24. Sklavos, N., Kaaniche, N.: On the design of secure primitives for real world applications. Microprocess. Microsyst. **80**, 103614 (2021)
25. Sujaudeen, N., Mirnalinee, T.T.: Tarnn: Task-aware autonomic resource management using neural networks in cloud environment. Concurr. Comput. Pract. Exp. **34**(8), e5463 (2022)
26. Tabuada, P.: Event-triggered real-time scheduling of stabilizing control tasks. IEEE Trans. Autom. Control **52**(9), 1680–1685 (2007)
27. Tang, Z., Zeng, A., Zhang, X., Yang, L., Li, K.: Dynamic memory-aware scheduling in spark computing environment. J. Parallel Distrib. Comput. **141**, 10–22 (2020)

28. Tong, Z., Chen, H., Deng, X., Li, K., Li, K.: A scheduling scheme in the cloud computing environment using deep q-learning. Inf. Sci. **512**, 1170–1191 (2020)
29. Tong, Z., Deng, X., Chen, H., Mei, J.: Ddmts: A novel dynamic load balancing scheduling scheme under sla constraints in cloud computing. J. Parallel Distrib. Comput. **149**, 138–148 (2021)
30. Wang, K., Fang, F., Da Costa, D.B., Ding, Z.: Sub-channel scheduling, task assignment, and power allocation for oma-based and noma-based mec systems. IEEE Trans. Commun. **69**(4), 2692–2708 (2020)
31. Wang, Z., Wang, G., Jin, X., Wang, X., Wang, J.: Caching-based task scheduling for edge computing in intelligent manufacturing. J. Supercomput. 1–23 (2022)
32. Xu, J., Zhang, Z., Hu, Z., Du, L., Cai, X.: A many-objective optimized task allocation scheduling model in cloud computing. Appl. Int. **51**(6), 3293–3310 (2021)
33. Xue, T., Chen, B., Wu, J., Wei, D., Freeman, W.T.: Video enhancement with task-oriented flow. Int. J. Comput. Vision **127**(8), 1106–1125 (2019)
34. Yao, W., Qi, N., Liu, Y., Xu, S., Du, D.: Homotopic approach for robot allocation optimization coupled with path constraints. IEEE Robot. Autom. Lett. **5**(1), 88–95 (2020)
35. Yuan, H., Tang, G., Li, X., Guo, D., Luo, L., Luo, X.: Online dispatching and fair scheduling of edge computing tasks: A learning-based approach. IEEE Intern. Things J. **8**(19), 14985–14998 (2021)
36. Zhang, J., Zhou, X., Ge, T., Wang, X., Hwang, T.: Joint task scheduling and containerizing for efficient edge computing. IEEE Trans. Parallel Distrib. Syst. **32**(8), 2086–2100 (2021)
37. Zhang, W.-Z., Elgendy, I.A., Hammad, M., Iliyasu, A.M., Du, X., Guizani, M., Abd El-Latif, A.A.: Secure and optimized load balancing for multitier iot and edge-cloud computing systems. IEEE Int. Things J. **8**(10), 8119–8132 (2021)
38. Zhu, M., Wang, A.I.: Model-driven game development: A literature review. ACM Comput. Surv. (CSUR) **52**(6), 1–32 (2019)