Computers, Materials & Continua
DOI: 10.32604/cmc.2023.035530
Article

Tech Science Press

Check for updates

Edge Computing Task Scheduling with Joint Blockchain and Task Caching in Industrial Internet

Yanping Chen[1,2,3], Xuyang Bai[1,2,3,*], Xiaomin Jin[1,2,3], Zhongmin Wang[1,2,3], Fengwei Wang[4] and Li Ling[4]

[1]School of Computer Science and Technology, Xi'an University of Posts and Telecommunications, Xi'an, 710121, China
[2]Shaanxi Key Laboratory of Network Data Analysis and Intelligent Processing, Xi'an, 710121, China
[3]Xi'an Key Laboratory of Big Data and Intelligent Computing, Xi'an, Shaanxi, 710121, China
[4]ZTE Corporation, Shenzhen, 51805, China
*Corresponding Author: Xuyang Bai. Email: xuyangbai1999@163.com
Received: 24 August 2022; Accepted: 14 December 2022

Abstract: Deploying task caching at edge servers has become an effective way to handle compute-intensive and latency-sensitive tasks on the industrial internet. However, how to select the task scheduling location to reduce task delay and cost while ensuring the data security and reliable communication of edge computing remains a challenge. To solve this problem, this paper establishes a task scheduling model with joint blockchain and task caching in the industrial internet and designs a novel blockchain-assisted caching mechanism to enhance system security. In this paper, the task scheduling problem, which couples the task scheduling decision, task caching decision, and blockchain reward, is formulated as the minimum weighted cost problem under delay constraints. This is a mixed integer nonlinear problem, which is proved to be nonconvex and NP-hard. To solve the optimal solution, this paper proposes a task scheduling strategy algorithm based on an improved genetic algorithm (IGA-TSPA) by improving the genetic algorithm initialization and mutation operations to reduce the size of the initial solution space and enhance the optimal solution convergence speed. In addition, an Improved Least Frequently Used algorithm is proposed to improve the content hit rate. Simulation results show that IGA-TSPA has a faster optimal solution-solving ability and shorter running time compared with the existing edge computing scheduling algorithms. The established task scheduling model not only saves 62.19% of system overhead consumption in comparison with local computing but also has great significance in protecting data security, reducing task processing delay, and reducing system cost.

Keywords: Edge computing; task scheduling; blockchain; task caching; industrial security

1 Introduction

In recent years, the world has started the fourth industrial revolution represented by the industrial internet, and the current manufacturing industry is developing in the direction of personalization, service, interconnection, intelligence, green, and ecology [1]. The industrial internet fully integrates modern technologies such as sensor networks and computer networks and has the advantages of low cost, low investment, and high practicability [2]. The development of the industrial internet has become a critical cornerstone for a country's transition to industrial power. However, due to the diversity of industrial devices, each system has its unique protocols, commands, and storage methods. Therefore, it is necessary to preprocess the data to communicate with each other [3]. Limited by the scarcity of resources such as CPU computing power, storage space, and energy capacity of industrial equipment, some complex computing tasks are inefficient locally [4]. To solve this problem, an effective solution is to use edge computing (EC). EC has a powerful computing ability that can increase the speed of processing tasks. Edge servers pre-process data from a variety of devices to make communication between devices easier. Compared with existing cloud computing frameworks, EC with high bandwidth and low delay can effectively fulfill the demand for latency in industrial production [5,6].

EC in industrial internet pays more attention to delay and security. Some tasks generated by equipment in intelligent factory manufacturing are high similarity such as face recognition, fire monitoring, judging the quality of solder joints, etc. The arrival times of these highly similar tasks are disordered, bringing a huge computational burden to the servers. Many scholars proposed that utilizing a caching mechanism to store dynamic data programs in EC systems can successfully reduce computational delay and energy consumption [7,8]. The inputs or outputs of computational tasks are cached, which can reduce the reuse of resources and the frequency of data transfers by invoking the data in the cache pool when the task similarity. With the help of simulation studies, reasonable deployment of task caching can reduce delay [9]. However, as the number of communication requests grows, the traditional centralized service approach will lose its ability to schedule in real-time. Task caching also puts more pressure on the central server. The server needs to consider the task caching results and then select the appropriate location in many servers to schedule the task. Furthermore, if the central server's scheduling and supervision are not timely, it is easy to produce a variety of issues connected to uncontrollable and unreliable data. If these problems are not solved, the processing time will be longer and data security will be affected.

As a result, security issues must be addressed as part of the development of task caching in EC. Blockchain, as a distributed and trusted technology, has been shown to possess many significant attributes, including security, non-tampering, and privacy [10,11]. Blockchain treats all edge servers as peer nodes and breaks the information transfer barrier between edge servers. The proposed blockchain task scheduling model with task caching can protect the security of scheduling data from the source. At the same time, task caching obscures the usage pattern and scheduling location to protect the user's privacy.

The motivation to incorporate blockchain into the caching mechanism is to ensure that heterogeneous edge devices can share secure and reliable cached data through the unique security mechanism of blockchain. Task caching provides low latency services for delay-intensive tasks, and edge servers provide abundant computing resources for computing-intensive tasks. Blockchain ensures reliable data communication sharing and the security of task scheduling. The speed of processing EC tasks will be significantly improved on the premise of a safe and reliable task cache. For such a complex framework,

this paper uses a global heuristic algorithm to select the task scheduling position, so that the task minimizes the cost within the allowable delay. The main contributions of this article are threefold:

(1) This paper establishes a joint task scheduling, task caching, and blockchain optimization model. Considering the characteristics of large amounts of tasks and high task similarity in the intelligent manufacturing scene of the industrial internet, we design a novel blockchain-assisted reliable caching mechanism for the industrial internet and propose an Improved Least Frequently Used (ILFU) algorithm to raise the content hit rate when scheduling the cache pool data.
(2) This paper combines blockchain with EC to address the challenges of data sharing insecurity and data leakage induced by distributed edge nodes. Through the unique security mechanism, blockchain provides reliable cached data for the EC cache pool, while ensuring distributed edge nodes achieve more secure information interaction. Blockchain is an extension of the edge cache pool that expands its cache capacity, allowing the system to match more types of tasks and minimize task processing time.
(3) Considering the impact of task caching and blockchain on task scheduling location, this paper proposes a task scheduling policy algorithm based on an improved genetic algorithm (IGA-TSPA), which effectively selects the appropriate task scheduling location according to delay constraints and resource constraints. Compared with the traditional EC scheduling algorithm, IGA-TSPA obtains excellent scheduling effects in complex data environments.

2 Related Works

Since the concept of EC is first presented, task scheduling has gotten a lot of attention. Du et al. considered processing resources, transmission power, and bandwidth allocation. A low-complexity suboptimal algorithm was proposed, and offloading decisions were solved by semidefinite relaxation and randomization [12]. Gao et al. investigated the problem of improper server load during the offload processing of EC. A two-stage computation scheduling strategy was proposed to minimize the delay [13]. Zhu et al. aimed to minimize task computation delay by jointly optimizing the non-orthogonal multiple access-based transmission duration and workload offloading allocation among edge computing servers, and then proposed the deep reinforcement learning-based algorithm to obtain the near-optimal offloading solution [14]. Wei et al. modeled task scheduling as a Markov process for unpredictable resources with energy in mobile EC [15]. Wang et al. proposed an integer particle swarm optimization method to solve the problems of offloading decisions, task assignment, and task order, which can improve resource efficiency and user satisfaction [16]. As the number of data and tasks increases, the available resources for EC will decrease. To solve this problem, some academics proposed task caching to further reduce latency. Zhou et al. presented a comprehensive optimization framework to minimize the service latency and adjust task offloading decisions at each time slot under the constraints of the device's energy and resource capacity to reduce the caching overhead [17]. Zhou et al. proposed an intelligent particle swarm optimization-based offloading strategy with the cache mechanism to find an appropriate offloading ratio to implement partial offloading [18]. Qian et al. proposed a joint push caching strategy based on hierarchical RL to cope with complex network tides to lessen the total amount of data transmission and maximize the utilization of bandwidth [19]. Wang et al. proposed an efficient cache algorithm to reduce the resource consumption of backhaul links by caching popular content on fog computing servers [20].

Task caching decreases the time of EC processing and is also significant for the development of EC. However, with distributed EC, maintaining safe task cache interaction appears to be a

challenge once again. In recent years, blockchain has developed rapidly, coinciding with EC because of its distributed characteristics, and EC is increasingly closely linked to the blockchain. The trust problem of computing-as-a-service scheduling in heterogeneous networks was addressed by Bai, who proposed a multilateral chain structure that can hold thousands of data and improves the efficiency of data on the chain. A two-stage Stackelberg game technique was presented to speed up the computational benefit of energy [21]. Liu et al. proposed caching cryptographic hashes on edge nodes for computationally intensive mining tasks as a solution to the proof-of-work problem in blockchain, using an alternating direction method of the multipliers-based algorithm to solve the offloading decision and caching strategy [22]. Luo et al. solved the problems of limited coverage in existing mobile EC and insufficient data supervision and tracking capability of service providers. It is proposed to use drones to dynamically cache data and upload the cached data to the blockchain to ensure data traceability without being tampered [23]. Liao et al. developed a secure intelligent task offloading framework to deal with vehicle fog computing task offloading delay under incomplete information while obtaining the lowest queuing delay and switching costs [24].

The above studies have achieved promising results in reducing delay, saving cost, and improving the quality of service for EC, but they are not perfect in ensuring the security of EC. While some research already combines EC and blockchain to protect data security, there are far too few studies concerning blockchain to protect EC caches. This paper combines blockchain with EC to improve the security of EC data cache and the security of smart manufacturing in the industrial internet. The work referenced in this article is compared in Table 1.

Table 1: Comparison of the work

Work	Task scheduling optimization target		Task caching	EC & Blockchain
	Delay & Cost	Other		
[12–16]	√	√		
[17–20]	√		√	
[21,24]	√			√
[22,23]			√	√

3 System Model

EC in the industrial internet pays more attention to latency and security than traditional EC [25]. Fig. 1 shows the scenario of smart manufacturing. The whole system is divided into three layers: the local layer, the edge layer and the cloud layer. The local device layer is a collection of manufacturing devices with limited computing and data collection capabilities. The devices are denoted as $N = \{1, 2, 3 \ldots n\}$. Tasks are generated at the device layer. Set the task attribute to $R_{ij} = \{r_{ij}, r_{wj}, r_{dj}, t_i^{max}\}$, r_{ij} represents the size of the data input for the task such as code or resources required, r_{wj} is the computational power required, r_{dj} is the index of cache results required by the task, and t_i^{max} is the maximum delay tolerated to complete the task. The edge layer is made up of various edge servers denoted as $M = \{1, 2, 3 \ldots m\}$. The edge servers not only process the task but also act as the miner node in the blockchain to perform the mining task. The edge server is responsible for updating the

newly generated results to the blockchain and ensuring the cache pool match more task types. Edge servers store and read cache data through smart contracts, and the query layer and contract layer are responsible for data processing in the blockchain. The third layer is the cloud layer, which has huge data storage capacity and computing capacity. Considering that the data in the blockchain will become more and more huge with the accumulation of time, the system can offload the data to the cloud. The completion of a task requires consideration of time constraints and computational power.

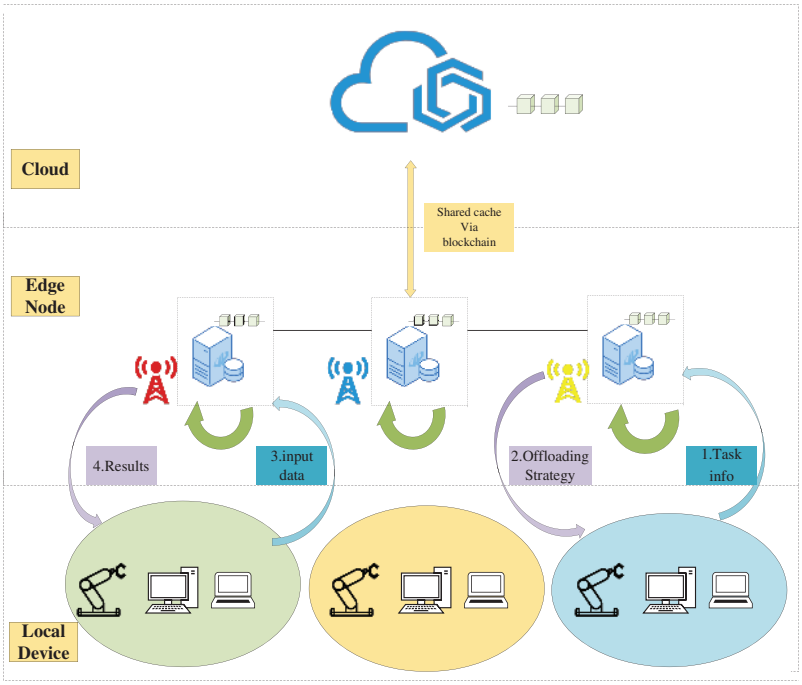


Figure 1: The edge cloud system with blockchain

Due to the limitation of energy and computing power of the local device, the system will schedule tasks to the edge server according to the optimal cost under the maximum allowable delay of tasks. In this paper, the task scheduling policy is defined as $x_{i,j} \in \{0, 1\}$, where $x_{i,j} = 0$ means the task is processed locally and $x_{i,j} = 1$ means the task is processed at edge servers. Blockchain provides reliable and strongly trusted data for edge cache pools.

3.1 Communication Model

When the task is scheduled to the edge server, the edge server needs sufficient resources. After the computation task is processed, the edge server will return the results. Set P_{ij} represents the uplink transmission capacity from the device i to edge server j, the uplink rate V_{ij} is

$$V_{ij} = B\log_2\left(1 + \frac{P_{ij}\,|h_i|\,|r_i|^{-a}}{\sigma^2}\right), \tag{1}$$

where σ^2 represents noise power, $|r_i|^{-a}$ represents path fading and $|h_i|$ represents the interaction between signals [22]. The upload time of the task is

$$T^{upload}\left(r_{ij}\right) = \frac{r_{ij}}{B\log_2(1 + P_{ij}\varpi)}, \tag{2}$$

where $\varpi = \frac{|h_i|\,|r_i|^{-a}}{\sigma^2}$. During the task upload process, the energy consumption of the local device is

$$E\left(e_1^l\right) = \frac{e_1^l r_{ij}}{\zeta B \log_2(1 + P_{ij}\varpi)}, \tag{3}$$

where ζ is the efficiency of the equipment transmission power amplifier. $E(e_0^l, e_1^l, e_{-1}^l)$ represents the local device energy consumption: e_0^l is the energy consumed by the device when the task is running locally, e_1^l is the energy consumed when the device transfers the task and e_{-1}^l is the energy consumed when the local device is idle.

3.2 Computing Model

In this paper, the scheduling locations of tasks are local devices and edge servers. When the task is executed local device, the task consumes on devices i is

$$E_0 = k(f_i^l)^2 r_{w,j}, \tag{4}$$

where $k = 10^{-26}$, and it is a coefficient depending on the chip architecture [26]. The task is scheduled to the edge server by considering the delay and cost. This paper establishes a model of edge caching and blockchain to provide task caching, presupposing that the cache pool data in the edge side server is $Y_i = (yc_1, yc_2 \ldots yc_n)$. The presence or absence of a task cache for task requirements in the cache pool has a different impact on system energy consumption. Then, this paper will discuss different cache states.

(1) Task cache in cache pool: When a task is generated, the data in the cache pool is retrieved by viewing the requirements of the task index r_{dj}. The retrieval time is set to $T^{select}(n)$ and n is the size of cache data. If the retrieved data exists in the cache pool, the system directly dispatches the cached results. Since the amount of data processed as a result is negligibly compared to the incoming data, the result return time is ignored [27]. The time for task processing is

$$T^1 = T_{ij}^{up} + T_{ij}^{select}, \tag{5}$$

where T_{ij}^{up} represents the upload time of the task between device i to edge servers j. And then, the energy consumption of the local device is

$$E_1^l\left(e_{-1}^l\right) = e_{-1}^l T_{ij}^{select} + E_{ij}^{up}. \tag{6}$$

(2) Task cache in the blockchain: Because of the limited cache storage in the edge servers, the system takes the blockchain as an extension of the task cache pool. If the edge node does not have the task results, the system can obtain the required task results from the blockchain, and the retrieval time is T^{find}. Therefore, the total task processing time at this point is

$$T^2 = T_{ij}^{up} + T^{find}, \tag{7}$$

The energy consumption of local devices is

$$E_2^l\left(e_{-1}^l\right) = e_{-1}^l T^{find} + E_{ij}^{up}. \tag{8}$$

(3) The other cache states: The system does not retrieve the required task results in the blockchain network or the edge cache pool before the deadline of the task. The task must be processed at edge servers. Set the computing power of the edge node is f_i^e. Therefore, the task processing time is

$$T_i^{edge} = \frac{r_{wj}}{f_i^e}, \tag{9}$$

The total processing time and the energy consumption is

$$T^3 = T_i^{edge} + T^{find} + T^{select}, \tag{10}$$

$$E_3^l = e_{-1}^l T^3 + E_{ij}^{up}. \tag{11}$$

3.3 Caching Mechanism

This paper proposes an ILFU algorithm for multiple repeated tasks due to the system's schedule policy is influenced by the amount of task cache resources available and the resource hit rate. The Least Frequently Used (LFU) algorithm selects the data with the lowest historical access frequency for elimination. The core idea of the LFU algorithm is that "if data has been accessed many times in the past, it will be accessed more frequently in the future". The LFU algorithm stores a large number of past accessed records. The past accessed records occupy a lot of space and are not necessarily used in the future, which results in lower cache utilization. In view of the problems in the LFU caching strategy, this paper maintains a time-interval task frequency table to record the number of cache task hits in a period of time and sets a threshold for cache hits. To match a variety of different types of tasks and ensure the data security of the cache pool, this paper requires that the task caching has only two update sources, one is the task result of node processing, and the other is the result stored in the blockchain.

The cache task is represented by $yc_i = \{r_{dj}, o_i, c_i, y_i\}$, r_{dj} represents the cache index, o_i represents the number of cache hits and c_i represents the space allocation, and define $y_i \in \{0, 1\}$ represents whether the result of task R_i exists in the cache pool or not. The total task capacity is limited by

$$\sum_{i=1}^{n} c_i y_i \leq C, \tag{12}$$

where C is the sum of caching storage units. If the cache space is enough, the new task results can be cached directly. The size of the task data that can be cached is expressed as

$$c_{i+1} \leq C - \sum_{i=1}^{n} c_i y_i. \tag{13}$$

If cache space capacity is insufficient, the system replaces the task with the smallest value o_i in the cache pool. Remove this task from the cache pool, and the size of task result data that can be cached is

$$c_{i+1} \leq C - \sum_{i=1}^{n} c_i y_i + MIN\ \ o_i. \tag{14}$$

In this paper, the time-interval task frequency table is established to record the task hit results in t time periods and initial threshold Ω. Assuming that the cache hit rate in period $[t-2n, t-n]$ is ψ, the system will determine the hit rate and the predetermined threshold at t time. The $\psi < \Omega$ means that the task distribution of the cache pool is not reasonable and the content of the cache pool needs to be updated. The system will update and import several tasks with high frequency in the time-interval

task frequency table from the blockchain network into the cache pool to generate new cache pool data. The specific process is described as Algorithm 1.

Algorithm 1: ILFU Algorithm

Input: A series of tasks: R; Time slices: t^p; Hit rate threshold: Ω; The size of task: c_i; Index of task requirements r_{dj}; Cache table: $O[n]$; The time-interval task frequency table: $H[task]$

Output: Cache table

1. **While** R **do**
2. **While** t^p **do**
3. **If** $Y[i] == r_{dj}$ // Tasks exist in the cache pool
4. $O[i] \leftarrow O[i] + 1$; $H[task[i]] \leftarrow H[task[i]] + 1$;
5. **If** $Y[\min c_j] \leq c_i$ // Tasks do not exist in the cache pool and cache resources are sufficient
6. $O[i] \leftarrow 1$; $H[task[i]] \leftarrow H[task[i]] + 1$; $C \leftarrow \sum_{i=1}^{n} c_i y[i]$; //Update time-interval task frequency table and Cache table
7. **Else** $Y[i]! = r_{dj}$ and $c_i <= C - \min c_j$ //Tasks do not exist in the cache pool and cache resources are insufficient
8. Select the minimum $Y[c_j]$ then $Y[\min c_j] \leftarrow c_i$; $O[i] = 0$; $H[task[i]] += 1$;
9. **End while**
10. **If** $\psi < \Omega$ **then**
11. Select the m tasks with the largest number of times selected in the time frequency table;
12. **End while**

3.4 Blockchain Mechanism

The blockchain network is the foundation for the security and reliability of the whole EC task caching, and it is a bridge to ensure the safe exchange of information among edge servers. It is also critical for the blockchain to update new tasks. To encourage edge nodes to update blockchain data, the system will reward hardworking miners. The probability of orphan blocks in blockchain networks is

$$P_{orghan} = 1 - e^{-\eta\theta(S_n)}, \tag{15}$$

where η is a fixed value, and $\theta(S_n)$ is a function that represents the size of the block. Therefore, the probability that miner successfully mines a block and its solution reaches consensus is

$$P = \mu_n e^{-\eta\theta(S_n)}, \tag{16}$$

where $\mu_n = \dfrac{P_n}{H}$, H is the total hash power of the blockchain network [28]. The whole reward of edge node mining new block is

$$R_n^{\min ing} = R\mu_n e^{-\eta\theta(S_n)}, \tag{17}$$

where R is reward to generate a new block.

3.5 Problem Formulation

Appropriate scheduling solutions are obtained by taking into account task delay limits, equipment computing resources, and storage resources. The total energy consumption of the whole system is

$$E^{all}(X, Y, Z) = \sum_{i=1}^{n} \left\{ \{(1 - x_i) E_0\} + \left\{ x_i \left\{ y_i E_1^l + (1 - y_i) \left[z_i E_2^l + (1 - z_i) E_3^l \right] + E^{up} \right\} \right\} \right\}. \tag{18}$$

The total cost of the system is obtained from the cost of blockchain mining reward and optimal scheduling decisions. Therefore, the cost of task scheduling is composed of formulas (17) and (18). The optimization problem in this paper can be summarized as

$$MIN\ F^{all}(X, Y, Z) = \varepsilon E^{all}(X, Y, Z) - (1 - z_i) R \mu_n e^{-\eta\theta(S_n)} \tag{19}$$

$$x_i \in \{0, 1\},\ y_i \in \{0, 1\},\ z_i \in \{0, 1\}, \forall i \in N \tag{20}$$

$$x_i \sum r_{r,j} \leq C_1, \forall i \in N \tag{21}$$

$$y_i \sum r_{w,j} \leq C_2, \forall i \in N \tag{22}$$

$$\sum f_i \leq F \tag{23}$$

$$(T^{off} + T^{mec}) \leq T^{deadline},\ z_i y_i (T^{find} + T^{off}) \leq T^{deadline} \tag{24}$$

where ε is the conversion coefficient between energy consumption and cost. Constraint 20 guarantees the task offloading, caching and blockchain cache is valid. Constraint 21 guarantees the task results cached do not exceed the caching capacity of the edge servers. Constraint 22 guarantees the task requirements offloaded do not exceed the computation resources of the edge servers. Constraint 23 guarantees the computing and power allocation is valid. Constraint 24 guarantees the time constraint.

4 IGA-TSPA

First, task scheduling is an NP-hard problem and gives proof of this problem.

Theorem 1: *The optimization problem is NP-hard*.

Proof of Theorem 1

The knapsack problem is a well-known NP-hard problem. Given N products of size v_i and profit p_i and V is the maximum capacity of the backpack, the objective is to find a set $s \in N$ that maximizes the profit of the backpack [19], mathematically, the knapsack problem is

$$\begin{aligned} &MAX \sum_{i \in s} p_i \\ &s.t \sum_{i \in s} v_i \leq V. \end{aligned} \tag{25}$$

This paper considers a special instance of our optimization problem, in which there are N tasks and the complete time is T^{all}. Tasks are executed locally or at edge servers, set t_i denotes the completion

time of task, and e_i denotes the scheduled energy consumption of the task r_i. Optimization problems can be expressed as

$$\begin{aligned} &MIN\frac{\sum_{i=0}^{i=n} e_i}{N} \\ &s.t\sum_{i=0}^{i=n} t_i \leq T^{all}. \end{aligned} \tag{26}$$

Redefined $w_i = -\frac{e_i}{N}$, the problem can be expressed as

$$\begin{aligned} &MIN\sum_{i=0}^{i=n} w_i \\ &s.t\sum_{i=0}^{i=n} t_i \leq T^{all} \end{aligned} \tag{27}$$

This special instance of the problem corresponds to the Knapsack problem with knapsack size $T^{all} = V$ and profit value $w_i = p_i$. Therefore, this problem is NP-hard. This completes the proof.

Task scheduling with task caching is a multi-constrained optimization problem, and the solution space of the task will be very large and complex as the tasks and the cached data increase. To solve this problem, this paper proposes IGA-TSPA for selecting the optimal feasible solution in the vast solution space. The genetic algorithm (GA) is a global heuristic algorithm, which follows the law of "superiority and inferiority" in nature to select feasible solutions and obtain the optimal solution to the problem [27]. However, the GA has disadvantages including large solution space, long iteration cycles, and slow convergence, which cannot meet the requirements of the industrial internet. Therefore, this paper improves the population initialization and variation of the GA to make the algorithm more suitable for the model of this paper.

4.1 Initialization

In IGA-TSPA, genes represent the calculation position of the task. The algorithm uses 0–1 binary encoding, where 0 represents that tasks are executed locally and 1 represents that tasks are processed at edge servers. In the initialization operation, IGA-TSPA first obtains the maximum time for local devices to process tasks according to task requirements and local device resources. Secondly, the threshold of the task is preset according to the requirement of the task resource. If it exceeds the preset threshold of the system, the system will set the initial value of the task to 1, representing that the task may be processed faster at the edge node. Otherwise, set it to 0. IGA-TSPA improves the speed of finding the optimal solution by reducing the size of the solution space.

4.2 Selection

The selection operation is the process of selecting individuals from the predecessor population to the next generation population. Individuals are generally selected based on the distribution of individual fitness [8]. The selection of the fitness function in this paper is determined by formula (19). The system will select individuals with a higher fitness function, using the roulette method, and the

probability of each individual being selected is proportional to the size of the value of the fitness function. The fitness f_i of an individual i, and the probability P_{i} of i being selected are given by $P_{\mathrm{i}} = \frac{f_i}{\sum_{i=1}^{GENE} f_i}$.

4.3 Crossover and Mutation

Crossover is the process of generating more superior offspring individuals through the selection of predecessor populations. To reduce the damage of crossover to the intended population, a single point of crossover is used in this paper. In GA, the mutation is the way to possibly skip local optimal solutions. Mutation operations can accelerate the convergence speed of optimal solutions while maintaining population diversity and preventing it fall into "premature maturity" [27]. Because the mutation is an uncontrollable, random operation, there are some cases of invalid operation, which instead make the value of the fitness function smaller than before.

This paper operates at the mutation as follows: Randomly select tasks in the task sequence as mutation points. First, determine whether the task needs to be mutated by the mutation probability Q_y;

Observe the resource requirements of the task at the mutation point, determine whether it exceeds the mutation threshold α_2, and decide whether to mutate at this point; The specific implementation steps of IGA-TSPA are summarized as Algorithm 2.

Algorithm 2: IGA-TSPA

Input: Population: po; Gene: ge; Number of iterations M; Probability of mutation: Q_y; Task initialization threshold: α_1; Mutation threshold: α_2;
Output: Global best position vector

$x_{i,j}^t \leftarrow random(0,1)$; //Initialization
  If ($R_n < \alpha_1$ and $x_{i,j}^t = 0$)
    $x_{i,j}^t \leftarrow 1$; //Further judge the initialization task based on threshold
While $num \leq M$
  For $i = 1$ to po do
    For $j = 1$ to ge do
      Calculate fitness and select individuals with high fitness according to formula (19);
      Single point of crossover;
      If ($x_{i,j}^t = 0 \&\& R_n > \alpha_2$)
        group[i][j] $\leftarrow 1$;
      Else group[i][j] $\leftarrow 0$; // Determine whether the task is a positive mutation
  End for
$num \leftarrow num + 1$;
End while
Return Global best position vector X;

5 Experiment and Analysis

To evaluate the performance of the proposed algorithm and model, this paper considers a decentralized edge node network. According to the actual situation, the cloud server has the strongest capacity, the edge server has the second strongest capacity and the terminal device has the weakest

capacity [29]. The computing frequency of the device and edge servers is 1 and 4 GHz respectively. The device node transmits at power 33 dBm on the channel with bandwidth 20 MHz and the noise power is −174 dBm/Hz. The static circuit power is set to 0.3 W. The device power when the task is calculated is set to 0.9 W [29]. The data size and the required number of CPU cycles of the task are randomly generated between [20, 40] Mb and $[0.1 \times 10^9, 1 \times 10^9]$ cycles [26]. The tolerance time of the task is randomly distributed between 0.1 and 5 s. The number of tasks is randomly distributed between 50 and 200.

This paper compares the scheduling policy solved by IGA-TSPA with the All-local offload policy where all tasks in the workflow are executed locally, the Full-edge offload policy where all tasks in the workflow are offloaded to the edge server, and the EC schedule policy with caching. If completely offloaded locally, tasks will be discarded due to high latency and failure to complete. The task processing in this paper is considered when the delay t_i^{max} is satisfied. The loss of tasks will incur huge system costs. Because blockchain-assisted cache exists in the system, the blockchain expands the cache pool and reduces the cost of edge servers in processing queuing latency and task computation. In Fig. 2, the overall cost of edge computing caching policy with blockchain is 37.81% of All-local, 72.64% of Full-edge, and 88.89% of edge computing offload policy with caching. Therefore, blockchain-assisted caching for EC has huge advantages in terms of cost reduction in task scheduling.

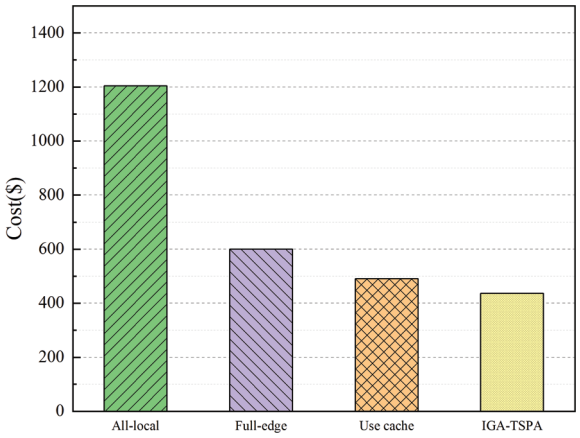
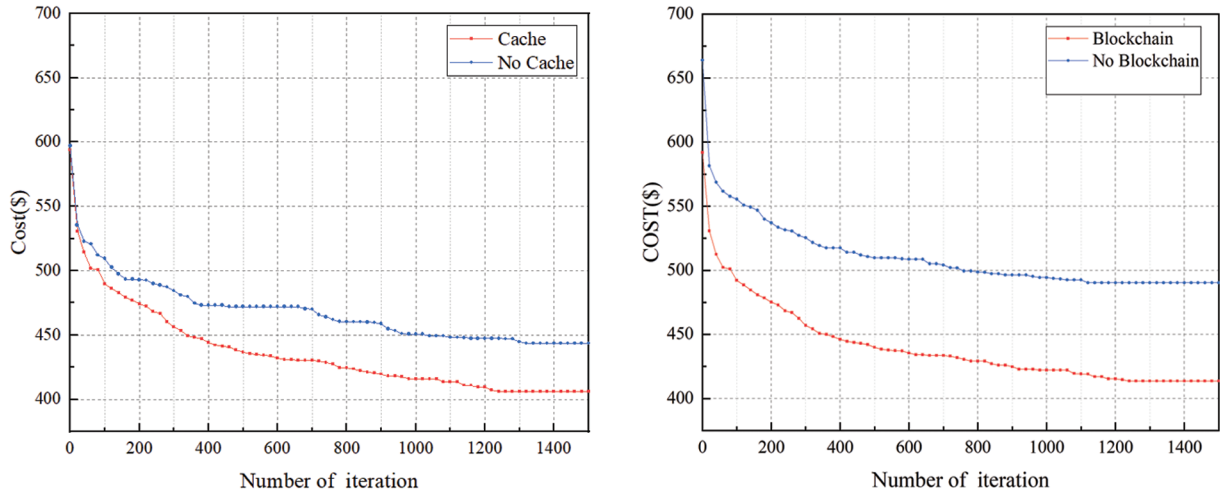


Figure 2: The simulation results of four strategies with different offloading methods

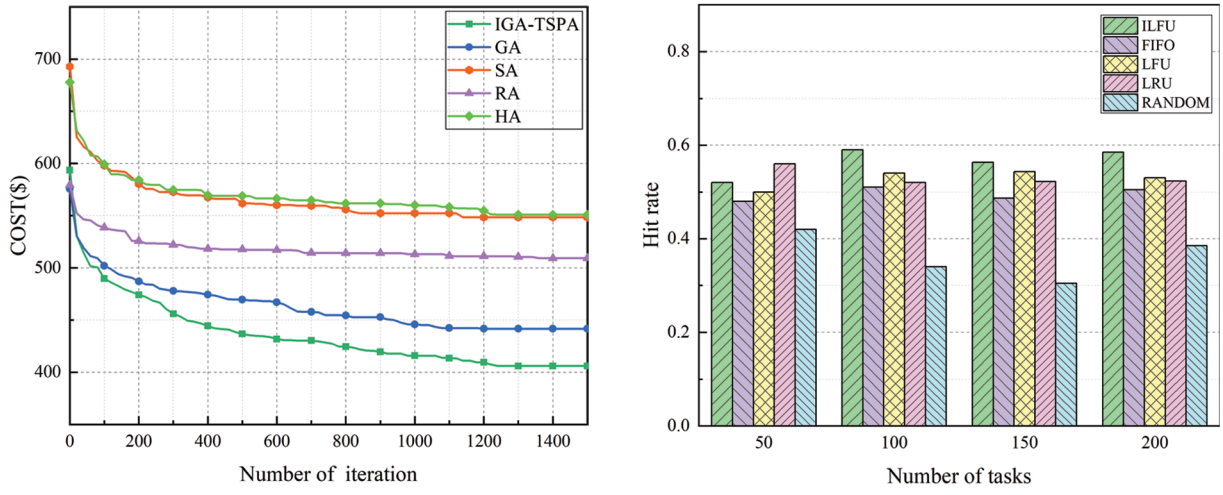
As shown in Fig. 3a, with task caching, both the convergence speed and the convergence performance of the algorithm are better than the case without caching. Without considering the return consumption of task results and the smaller cache search time, the task upload cost and the edge computation cost affect the whole cost of the system. Especially in industrial scenes, task caching can reduce the EC cost. The size of the cache space determines whether the cache type is complete. The blockchain expands the cache space, so the cost of the system will decrease. Ignoring the retrieval cost, since the blockchain has a larger cache capacity than the EC cache pool to match more task types, the no-cache curve in Fig. 3a drops more significantly than the curve cost in Fig. 3b. This model can meet the requirements of industrial internet for delay and cost.



(a) Cache and no cache (b) Blockchain cache and no blockchain cache

Figure 3: Simulation results of different models

This paper compares the ILFU algorithm with the existing classical caching algorithms including FIFO (First Input First Output), LRU (The Least Recently Used), LFU, and random algorithm. In Fig. 4a fixing the cache pool size, it can be seen that the cost after convergence is smaller than the other four algorithms for the same number of iterations. In Fig. 4b, as the increase of number of multiple repeated tasks, the hit rate of ILFU and LFU gradually increases, especially when the number of tasks reaches 200. Because the ILFU algorithm improves the LFU to prevent the task with the most frequent cache from occupying the cache for a long time. Therefore, the ILFU task cache hit rate is higher than other algorithms.

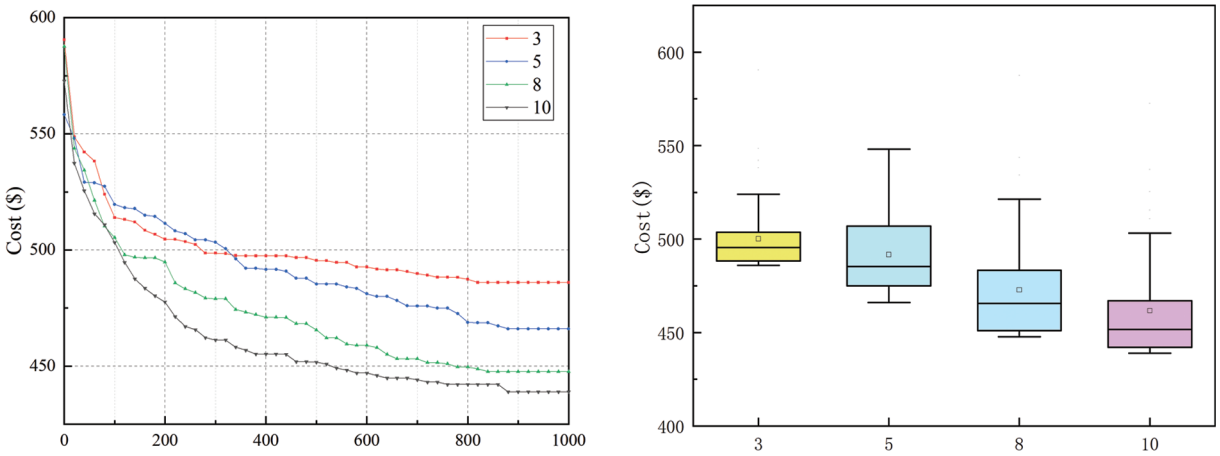


(a) Cost (b) Hit rate

Figure 4: The simulation results in different cache capacity

In Figs. 5a and 5b, the experiment maintains the number of tasks and increases the cache pool capacity. And then the cost reduction caused by the expansion of the cache is relatively large with the

expansion of cache space from 3 to 8 Gbit, but if the cache space is expanded from 8 to 10 Gbit, the system cost reduction rate gradually becomes smaller. In the process of task scheduling, some tasks are still executed locally. As a result, when the cache pool capacity reaches a certain level, the system cost will not be affected by the cache space because the task type is limited. As can be seen from Fig. 5b, the cost continues to decline as the cache space grows, but the distribution of the minimum cost varies greatly due to the different task types.

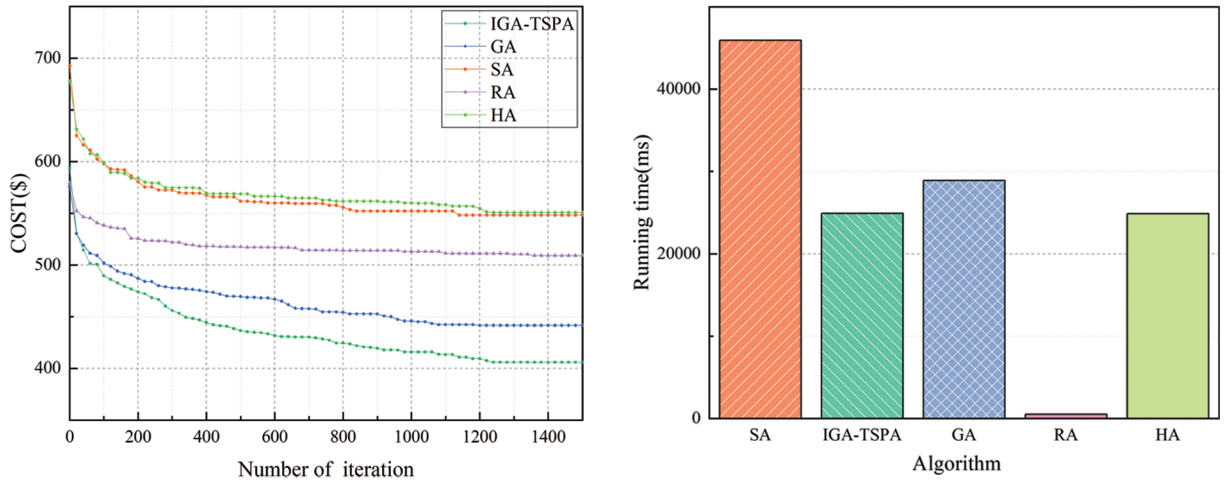


(a) Cost convergence under different cache capacity (b) Cost distribution under different cache capacity

Figure 5: The simulation results of cost with different cache capacity

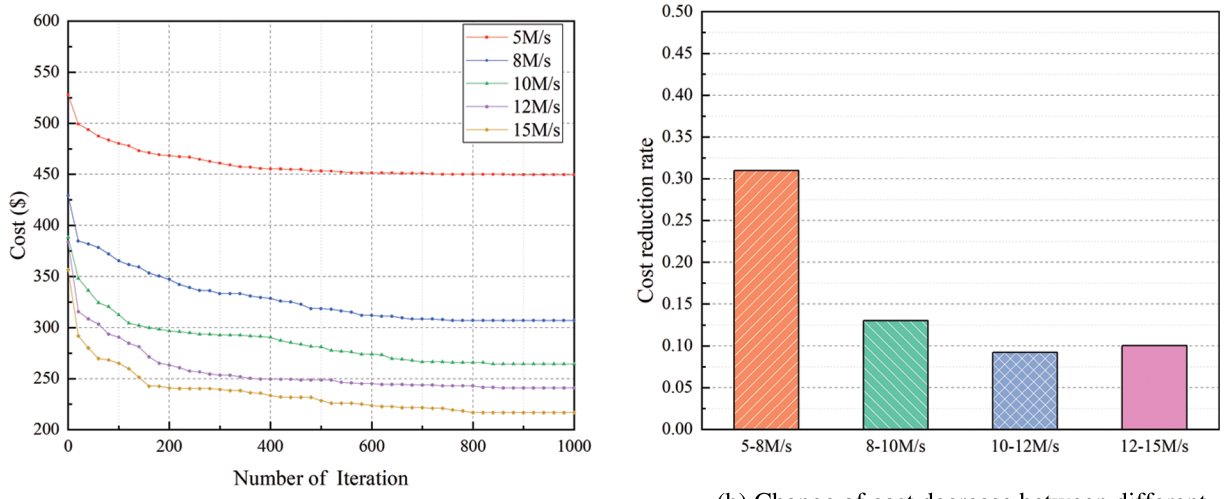
Based on the excellent caching strategy, this paper analyzes the performance of IGA-TSPA. As shown in Fig. 6a, this paper compares IGA-TSPA with several common global search algorithms such as the Genetic algorithm (GA), Simulated Annealing algorithm (SA), Hill-Climbing algorithm (HA), and Random algorithm (RA). The IGA-TSPA can obtain good convergence results. The overall cost of IGA-TSPA is about 88% of the Genetic Algorithm. Due to the GA randomly generating a large number of solutions, the solution space is large and it is not easy to search for the optimal solution. During the process of initialization, the IGA-TSPA limits the initialization of some tasks and reduces the time in selection. At the same time, in the mutation process, the IGA-TSPA avoids some useless mutations and improves the convergence effect of the algorithm by limiting the mutation operation. As depicted in Fig. 6b, this paper compares the running time of the different algorithms. From the histogram, the running time of the random algorithm is the shortest, and the running time of GA and IGA-TSPA is moderate. Moreover, because the IGA-TSPA reduces the number of feasible solutions in the initialization, the algorithm gets the optimal value faster, which proves that the improved algorithm is effective.

To further verify the advantages of the algorithm, this paper uses the control variable method to simulate the factors that affect the system cost, such as communication transmission rate. As shown in Fig. 7, the preset communication upload rates are 5, 8, 10, 12, and 15 M/s, respectively. With the increase in communication rate, more and more tasks are uploaded to the edge servers, and the effect of cost control is becoming more apparent. On the contrary, with the increase in communication rate, the cost decline rate tends to be more and more gentle, and the decline rate remains between 10% and 15% because some tasks are still being processed locally in the process of task scheduling. Therefore, in the model proposed in this paper, the increase in communication rate has an impact on the reduction of industrial costs.



(a) Performance of scheduling algorithms (b) Running time of scheduling algorithms

Figure 6: Performance comparison of different scheduling algorithms



(a) Cost convergence under different upload rates (b) Change of cost decrease between different upload rates

Figure 7: The simulation results of cost change based on different upload rates

6 Conclusion

To solve the task scheduling problem caused by a large amount of data, complex task types, and strong heterogeneity of equipment in a modern industrial environment in this paper. This paper establishes a joint optimization model of task caching, blockchain, and task scheduling. And then a three-layer architecture of industrial devices, edge computing, and cloud computing layers with blockchain technology is proposed to ensure communication and data storage security. The combination of blockchain and cache mechanism greatly improves the data security of the industrial internet, and its effectiveness in reducing delay and cost. Meanwhile, the ILFU algorithm is proposed to improve the content hit rate of the edge computing cache pool and further reduce the task delay. Finally, based on ensuring the security of cache and data sharing, considering the influence of task

cache, blockchain, and blockchain reward on scheduling policy, this paper proposes IGA-TSPA to optimize the cost problem of task processing. In the case of guaranteeing the delay, IGA-TSPA can obtain the scheduling policy with minimum cost. The large-scale simulation experiments have proved the effectiveness of the model and algorithm. In the future, we will conduct more in-depth research on blockchain technology to make it more suitable for edge computing in access control and privacy security.

Acknowledgement: This work was supported by the Communication Soft Science Program of Ministry of Industry and Information Technology of China (No. 2022-R-43), the Natural Science Basic Research Program of Shaanxi (No. 2021JQ-719), Graduate Innovation Fund of Xi'an University of Posts and Telecommunications (No. CXJJZL2021014). The Youth Innovation Team of Shaanxi Universities "Industrial Big Data Analysis and Intelligent Processing", and the Special Funds for Construction of Key Disciplines in Universities in Shaanxi.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] N. Shariatzadeh, T. Lundholm, L. Lindberg and G. Sivard, "Integration of digital factory with smart factory based on internet of things," *Procedia Cirp*, vol. 50, no. 1, pp. 512–517, 2016.
[2] Y. F. Wang and W. Dai, "Industrial internet of things: Ecological entrance ticket for future manufacturing industry," *China Industry Review*, vol. 2017, no. 4, pp. 28–34, 2017.
[3] J. Gao, X. G. Yan and B. Liang, "Research on IIoT architecture of edge computing based on blockchain," *Computer Application Research*, vol. 37, no. 7, pp. 2160–2166, 2020.
[4] H. Wang, Y. Li, X. Zhao and F. Yang, "An algorithm based on markov chain to improve edge cache hit ratio for blockchain-enabled IoT," *China Communications*, vol. 17, no. 9, pp. 66–76, 2020.
[5] D. Georgakopoulos, P. P. Jayaraman, M. Fazia, M. Villari and R. Ranjan, "Internet of things and edge cloud computing roadmap for manufacturing," *IEEE Cloud Computing*, vol. 3, no. 4, pp. 66–73, 2016.
[6] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.
[7] S. Wang, X. Zhang, Y. Zhang, J. Yang and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, no. 1, pp. 6757–6779, 2017.
[8] Z. Wang, G. Wang, X. Jin, X. Wang and J. Wang, "Caching-based task scheduling for edge computing in intelligent manufacturing," *The Journal of Supercomputing*, vol. 78, no. 4, pp. 1–23, 2021.
[9] Y. Miao, Y. Hao, M. Chen, H. Gharavi and K. Hwang, "Intelligent task caching in edge cloud via bandit learning," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 1, pp. 625–637, 2021.
[10] R. Zhang, R. Xue and L. Liu, "Security and privacy on blockchain," *ACM Computing Surveys*, vol. 52, no. 3, pp. 1–34, 2019.
[11] A. Dorri, M. Steger, S. S. Kanhere and R. Jurdak, "Blockchain: A distributed solution to automotive security and privacy," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 119–125, 2017.
[12] J. Du, L. Zhao, J. Feng and X. Chu, "Computation offloading and resource allocation in mixed Fog/Cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1594–1608, 2018.
[13] M. Gao, R. Shen, J. Li, S. Yan, Y. Li *et al.,* "Computation offloading with instantaneous load billing for mobile edge computing," *IEEE Transactions on Services Computing*, 2020, https://doi.org/10.1109/TSC.2020.2996764.

[14] B. Zhu, K. Chi, J. Liu, K. Yu, and S. Mumtaz, "Efficient offloading for minimizing task computation delay of NOMA-based multiaccess edge computing," *IEEE Transactions on Communications*, vol. 70, no. 5, pp. 3186–3203, 2022.
[15] Z. Wei, B. Zhao, J. Su and X. Lu, "Dynamic edge computation offloading for internet of things with energy harvesting: A learning method," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4436–4447, 2019.
[16] B. Wang, J. Cheng, J. Cao, C. Wang, and W. Huang, "Integer particle swarm optimization based task scheduling for device-edge-cloud cooperative computing to improve SLA satisfaction," *PeerJ Computer Science*, vol. 8, no. e893, pp. 1–22, 2022.
[17] R. Zhou, X. Wu, H. Tan and R. Zhang, "Two time-scale joint service caching and task offloading for UAV-assisted mobile edge computing," in *IEEE INFOCOM 2022-IEEE Conf. on Computer Communications*, London, United Kingdom, pp. 1189–1198, 2022.
[18] W. Zhou, L. Chen, S. Tang, L. Lai, J. Xia *et al.,* "Offloading strategy with PSO for mobile edge computing based on cache mechanism," *Cluster Computing*, vol. 25, no. 4, pp. 2389–2401, 2022.
[19] Y. Qian, R. Wang, J. Wu, B. Tan and H. Ren, "Reinforcement learning-based optimal computing and caching in mobile edge network," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2343–2355, 2020.
[20] K. Wang, J. Li, Y. Yang, W. Chen and L. Hanzo, "Content-centric heterogeneous fog networks relying on energy efficiency optimization," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13579–13592, 2020.
[21] F. Bai, T. Shen, Z. Yu, K. Zeng and B. Gong, "Trustworthy blockchain-empowered collaborative edge computing-as-a-service scheduling and data sharing in the IIoE," *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 14752–14766, 2022.
[22] M. Liu, F. R. Yu, Y. Teng, V. C. Leung, and M. Song, "Computation offloading and content caching in wireless blockchain networks with mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 11008–110021, 2018.
[23] S. Luo, H. Li, Z. Wen, B. Qian, G. Morgan *et al.,* "Blockchain-based task offloading in drone-aided mobile edge computing," *IEEE Network*, vol. 35, no. 1, pp. 124–129, 2021.
[24] H. Liao, Y. Mu, Z. Zhou, M. Sun, Z. Wang *et al.,* "Blockchain and learning-based secure and intelligent task offloading for vehicular fog computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4051–4063, 2020.
[25] X. Jin, W. Hua, Z. Wang and Y. Chen, "A survey of research on computation offloading in mobile cloud computing," *Wireless Networks*, vol. 28, no. 4, pp. 1563–1585, 2020.
[26] J. Zhang, X. Hu, Z. Ning, E. C. H. Ngai, L. Zhou *et al.,* "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2633–2645, 2018.
[27] A. Zhu and Y. Wen, "Computing offloading strategy using improved genetic algorithm in mobile edge computing system," *Journal of Grid Computing*, vol. 19, no. 3, pp. 1–12, 2021.
[28] Z. Xiong, S. Feng and D. Niyato, P. Wang and Z. Han, "Optimal pricing-based edge computing resource management in mobile blockchain," in *2018 IEEE Int. Conf. on Communications. ICC*, Kansas City, MO, USA, pp. 1–6, 2018.
[29] H. Wu, K. Wolter, P. Jiao, Y. Deng, Y. Zhao *et al.,* "EEDTO: An energy-efficient dynamic task offloading algorithm for blockchain-enabled IoT-edge-cloud orchestrated computing," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2163–2176, 2021.