Security Enhanced Edge Computing Task Scheduling Method Based on Blockchain and Task Cache

Cong Li
The Information Engineering Institute, Yellow River Conservancy Technical Institute, Kaifeng, 475004, China

***Abstract*—Aiming at edge computing nodes' limited computing and storage capacity, a two-layer task scheduling model based on blockchain and task cache was proposed. The high-similarity task results were cached in the edge cache pool, and the blockchain-assisted task caching model was combined to enhance system security. The genetic evolution algorithm was used to solve the minimum cost that the optimal scheduling model can obtain. The genetic algorithm's initialization and mutation operations were adjusted to improve the convergence rate. Compared with algorithms without cache pooling and blockchain, the proposed joint blockchain and task caching task scheduling model reduced the cost by 9.4% and 14.3%, respectively. As the capacity space of the cache pool increased, the system cost gradually decreased. Compared with the capacity space of 3GB, the system cost of 10Gbit capacity space was reduced by 10.6%. The system cost decreased as the computing power of edge nodes increased. Compared with edge nodes with a computing frequency of 8GHz, the nodes cost at 18GHz was reduced by 36.4%. Therefore, the proposed edge computing task scheduling model ensures the security of task scheduling based on reducing delay and control costs, providing a foundation for modern industrial task scheduling.**

Keywords*—Blockchain; task cache; edge computing; task scheduling; industrial internet

I. INTRODUCTION

With the developing 5G communication and the Internet of Things, many intelligent devices with computing power are widely used in industrial automation systems. The rapid increase of intelligent equipment in factories leads to the explosive growth of industrial Internet data [1]. Due to local devices' limited computing resources and storage capacity, some resource-intensive tasks will be scheduled to cloud servers for processing. However, when cloud servers are deployed far from local devices, the interaction between tasks and data can result in significant latency, posing a risk of data leakage and attack [2-4]. The introduction of edge computing into the industrial Internet can satisfy the real-time demand, security, and reliability in the industry. By scheduling tasks to the edge, industrial equipment has sufficient resources to handle more complex tasks [5-7]. When the resource results of a task have high similarity, deploying task caching can reduce repeated resource calls [8]. Gao et al. proposed a case layer solution of joint unloading scheduling and resource allocation to reduce task delay and energy consumption in on-board edge computing, combining deep Q network and gradient descent method. This algorithm effectively reduced latency and energy consumption [9]. Chen et al. built an edge computing container deployment model for the delay-sensitive problem of tasks. Through ameliorating the initialization, crossover, and other operations of Genetic Algorithm (GA), the optimal deployment of containers was achieved. Compared with traditional algorithms, the deployment cost of this model was reduced by 22% [10]. Although edge computing and task caching have many advantages, the diversity of servers brings an additional burden to task scheduling.

Blockchain is a distributed accounting technology that integrates multiple technologies such as distributed storage, cryptography, and consensus mechanisms. It can ensure data security, tamper resistance and privacy, and can well match distributed edge computing [11-13]. The introduction of blockchain into a cloud-edge-end three-layer architecture can enhance industrial security. The collaboration between edge servers and blockchain enables secure data transmission and reliable storage, reducing computational costs and latency [14-16]. Rivera A V et al. proposed a secure task-sharing blockchain framework to enhance user experience. They set up a trusted cooperation mechanism in multi-access edge computing and designed cooperation incentives to speed up computing. This framework ensured trust between servers and enabled real-time task sharing [17]. Zhang H et al. put forward a mobility management scheme using blockchain to address the security of unloading tasks. They used Lyapunov optimization algorithm, combined with base station wireless handover and service migration decisions, to achieve dynamic optimization of the target. This solution effectively reduced the latency and failure rate of computing tasks [18]. Chen J et al. proposed a decentralized management scheme based on blockchain to address the transparency in collaboration benefits. They used diligent proof and delegated diligent proof consensus mechanisms, combined with sequential decision-making and Byzantine fault-tolerant algorithms, to improve the time-sensitivity of edge collaboration. This scheme had high security and fault tolerance [19]. Liu R et al. proposed a trusted data storage mechanism using blockchain to address data management security in the industry. Combining sharding and a two-layer Merkle tree structure, they utilized random low-density parity correction code encoding to reduce storage pressure on lightweight nodes. This mechanism effectively reduced the network load of nodes and improved data storage security [20]. Li G et al. proposed edge bandwidth and storage optimization algorithms to overcome network overload caused by distributed transmission. They built a dynamic blockchain and combined it with a network simulator to construct blockchain. This algorithm improved both transmission bandwidth efficiency and blockchain construction efficiency [21].

The computing power and storage capacity of edge servers are stronger than those of local devices, and edge computing has lower task latency. With the explosion of industrial task data, the resources of edge servers are gradually scarce, and the task scheduling problem of edge computing is NP hard. Although there are many researches on edge computing task scheduling, there are some problems. For multi-objective optimization problems, most studies are based on the optimization objectives of time delay and energy consumption to develop task schedules. For complex and repetitive tasks in the industrial Internet, task similarity should be taken into account to improve edge computing capability and efficiency. For data security issues, most studies mainly focus on data security and privacy, with little consideration given to the latency and energy consumption caused by them. Delay and energy consumption should be included, and the best outcome plan should be balanced between performance and security levels. Therefore, the research will combine the blockchain, edge computing and task cache, and propose a security enhanced edge computing task scheduling method based on the blockchain and task cache. On the basis of ensuring data security, appropriate task scheduling strategies will be developed to give full play to various technical advantages to meet the demand for delay, cost and security in the industrial scenario. Faced with the enormous security and resource pressures brought by massive data on industrial equipment, an optimization model is established to ensure the secure scheduling of tasks to the greatest extent possible. Considering the problem of high task similarity in industrial Internet, the research adopts the improved least access frequency algorithm to improve the hit rate of cache content. It combines block chain technology with edge computing to solve the problem of data leakage at edge nodes. As a supplement to the cache pool, blockchain increases task cache capacity and reduces task processing time. Therefore, the task scheduling method proposed in the study provides technical support for industrial task scheduling.

II. Methods and Materials

To reduce the delay and cost of edge computing in the industrial Internet, this research combines task caching and blockchain to design new algorithms to improve the hit rate of task caching. Blockchain and task cache are introduced into edge computing and combined with task scheduling strategy to improve the processing efficiency of tasks, reduce costs, and achieve reliable data storage and low overhead of task scheduling.

A. Blockchain-Based Edge Caching Model

The rapid development of intelligent devices makes the industrial Internet have higher requirements for resources. Due to local devices' limited battery, storage, and computing resources, offloading computing tasks to the cloud layer can cause additional network latency and bandwidth consumption. Therefore, adding an edge server layer between the cloud layer and the local device layer can alleviate the burden on the cloud network. Fig. 1 shows a blockchain-based cloud-edge-end system, which is divided into three layers, including cloud servers, edge servers, and local devices.

The local device layer is composed of industrial equipment and has minimal computing and information organization capabilities. The edge server layer is composed of edge devices with strong computing and caching capabilities, which can be used for mining in blockchain networks. The mining process consists of three steps: (1) Edge nodes add cached results to the blockchain and increase the task type of the cache pool. (2) When the task is scheduled to the edge server, this structure searches a cache pool and returns this result directly if it is found. If it is not found, it performs calculations locally and updates the cache pool and blockchain data. (3) For every new task type added to the edge node, it can receive corresponding rewards, increasing the interest of the edge server in mining. A cloud server layer owns powerful data storage and computing capabilities. When the total data in the blockchain network exceed its capacity, this system can upload some data requests to the cloud. These data in the industrial Internet are highly similar and have a large number. Storing relevant task data into the cache pool can speed up data processing. To ensure the cache pool security, the data source can only be task results processed by edge nodes and stored results in the blockchain. Fig. 2 shows the cache pool's data source.

The cache hit rate, task complexity, and promotion performance are important factors to measure the effectiveness of cache strategies, and cache strategies should be selected according to actual scenario needs. Based on the changes in tasks in industrial production, the Least Frequently Used (LFU) algorithm in traditional caching strategies is improved. A Task Caching of Improved LFU (TC-ILFU) algorithm is put forward to elevate the cache hit rate. The core of LFU is to prioritize the elimination of cache data with the lowest frequency of use. Fig. 3 shows the main idea of ILFU-TC. This algorithm establishes a time task frequency table, records the hit rate data of cached task content over a period of time, and determines whether new tasks can be added to the cache pool based on the hit rate.

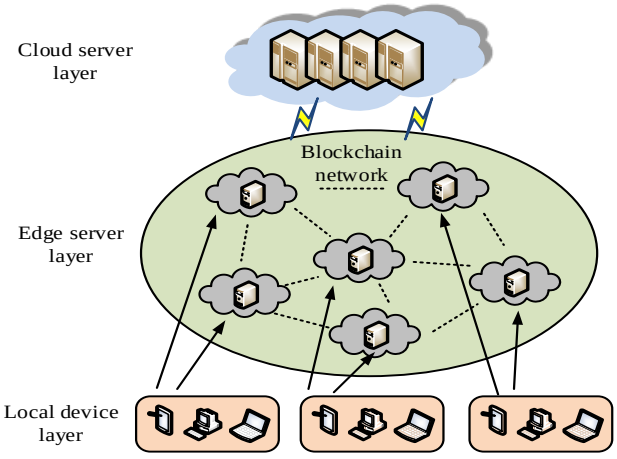


Fig. 1. Cloud-edge-end scheduling system based on blockchain.

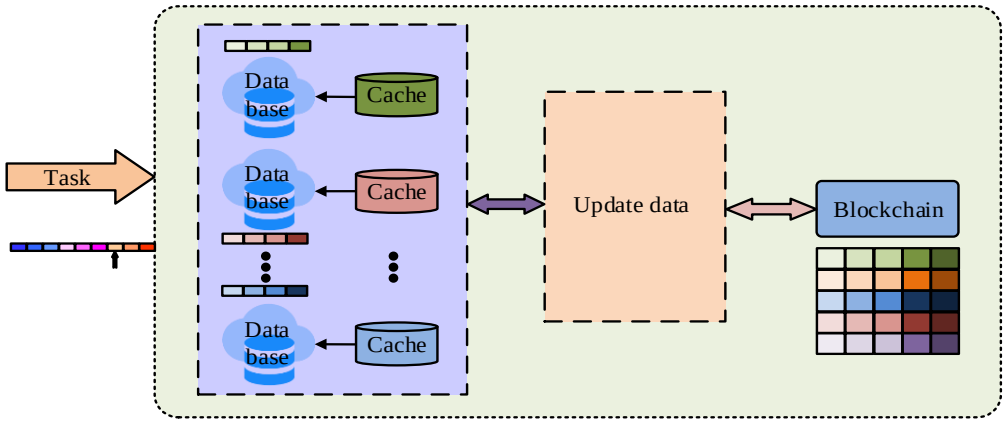


Fig. 2. Cache pool data source.

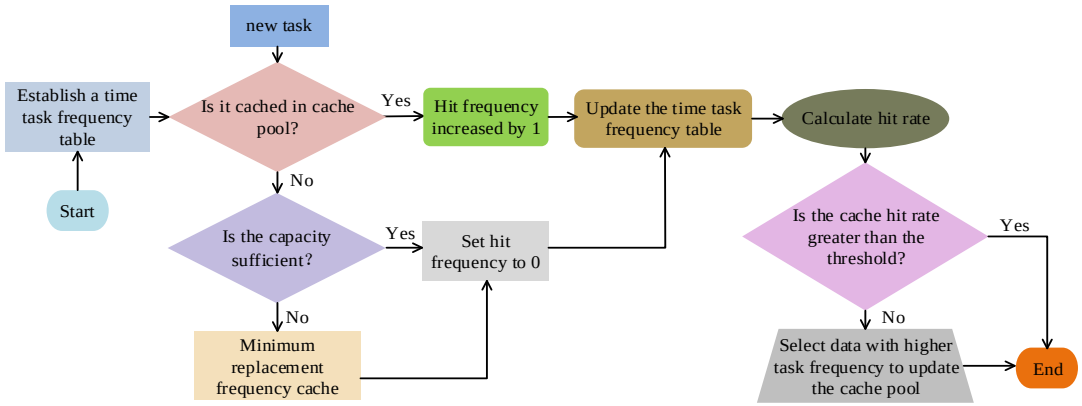


Fig. 3. The main ideas of improved LFU.

The time task frequency table records the task requirements for a period of time. When a new task appears in the edge cache pool, the first step is to determine whether it has been cached. For tasks belonging to the cache pool, the hit frequency is directly increased by 1, and the time task frequency table is updated. If the task has not appeared in the cache pool, it is determined whether there is sufficient space and time based on the cache pool capacity. If the cache capacity space is sufficient, the hit frequency is set to 0, and the time task frequency table is updated. If the cache capacity space is insufficient, the system will replace the task with the minimum number of hits in the cache pool with the new task. At this point, the hit frequency is set to 0, and the time task frequency table is updated. The system calculates the hit rate during this period and compares it with the set cache threshold. If the hit rate is greater than this cache threshold, the task content cache matches the task's characteristics. If the hit rate is less than the cache threshold, this system will select data with higher task frequency from the task frequency table at that time and import it into the cache pool to update the cache pool data. The blockchain network can bring safe and reliable cache data to edge computing and ensure that edge nodes can safely exchange information. In blockchain, the probability of orphan blocks being generated is represented by Eq. (1).

$$P_{orp} = 1 - e^{-\eta\lambda(S_n)} \tag{1}$$

In Eq. (1), P_{orp} represents the probability of orphan blocks. η is a fixed value, $\eta = 1/600$. $\lambda(S_n)$ is a function of block size. The probability of generating new task blocks is represented by Eq. (2).

$$P_{new} = \frac{P_n}{H} e^{-\eta\lambda(S_n)} \tag{2}$$

In Eq. (2), P_{new} represents the probability of a new task block. P_n is the hashing capability of blockchain networks. H is the blockchain network's hash power. The reward for mining new task blocks at edge nodes is represented by Eq. (3).

$$R_n^{rew} = R\mu_n e^{-\eta\lambda(S_n)} \tag{3}$$

In Eq. (3), R_n^{rew} represents the mining reward. R is a task set.

B. Edge Computing Task Scheduling Algorithm

Due to local devices' limited battery capacity and computing resources, tasks are scheduled to edge servers for processing within the maximum latency allowed range. The local device generates tasks, and the task scheduling strategy follows constraints represented by Eq. (4).

$$x_{ij} \in \{0,1\}, i \in N, j \in M \tag{4}$$

In Eq. (4), x_{ij} is the scheduling location of the task.

$x_{ij}=1$ is the calculation of task i on edge server j. $x_{ij}=0$ means the processing of tasks on local devices. N is the total local device. M is the total edge server. Blockchain is a decentralized distributed storage ledger with high security and certainty. The limited capacity of the cache pool requires a diverse range of task types to be cached, ensuring that task scheduling has multiple selectivity. According to the task scheduling strategy, when a task is calculated on an edge server, this server needs sufficient computing power to process the task and return this result to the device. The uplink rate of local device scheduling tasks to edge servers is approximated using Shannon's formula, represented by Eq. (5).

$$V_{ij}=B\log_2\left(1+P_{ij}\varpi\right)=B\log_2\left(1+\frac{P_{ij}\left|h_i\right|\left|r_i\right|^{-a}}{\sigma^2}\right) \tag{5}$$

In Eq. (5), V_{ij} represents the transmission rate of local device i scheduling tasks to edge server j. B is the bandwidth. ϖ is the signal-to-noise ratio. P_{ij} is the transmission capacity of the device i. $|h_i|$ is channel interference. $|r_i|^{-a}$ is path decay. σ^2 is the power of Gaussian white noise. According to the task attributes, the upstream time for scheduling tasks from local devices to edge servers is represented by Eq. (6).

$$T_{ij}=\frac{r_{ij}}{V_{ij}}=\frac{r_{ij}}{B\log_2\left(1+P_{ij}\varpi\right)} \tag{6}$$

In Eq. (6), T_{ij} represents the transmission delay from the local device scheduling task to the edge server. r_{ij} is the local device's task data scale. The upload energy consumption of local devices is represented by Eq. (7).

$$E_{up}=\frac{e_1^l r_{ij}}{\zeta B\log_2\left(1+P_{ij}\varpi\right)} \tag{7}$$

In Eq. (7), E_{up} represents the upload energy consumption of the local device. ζ is the transmission amplifier efficiency of the device. Local device consumption is $E\left(e_{-1}^l,e_1^l,e_0^l\right)$. e_{-1}^l is idle local devices' energy consumption. e_0^l represents local computing tasks' energy consumption. e_1^l means local device transmission's energy consumption. The task is represented by Eq. (8) when processing on the local device.

$$T_i^l=\frac{c_i}{f_i^l} \tag{8}$$

In Eq. (8), T_i^l represents the calculation delay of the local device. c_i is the number of chips. f_i^l represents the computing power of local devices. The energy consumption of terminal devices for processing tasks is represented by Eq. (9).

$$E_0=\kappa\left(f_i^l\right)^2 r_{wj} \tag{9}$$

In Eq. (9), E_0 represents the terminal calculation energy consumption. κ is a chip architecture coefficient. r_{wj} is the demand for computing power. Local device generates tasks. According to the task scheduling strategy, the tasks are offloaded to the edge server for computation, and edge caching and blockchain based edge caching models are established. When there is content cache in the cache pool, retrieve cached data according to task indexing requirements. When data exist, the system directly distributes cached results. The task processing time is represented by Eq. (10).

$$T_1=T_{ij}+T^{select}\left(n\right) \tag{10}$$

In Eq. (10), T_1 represents the processing time of the task in the cache pool. $T^{select}\left(n\right)$ means data retrieval time. n is the cache data size. The energy consumption of the terminal server is represented by Eq. (11).

$$E_1=E_{up}+e_{-1}^l T^{select}(n) \tag{11}$$

In Eq. (11), E_1 means the terminal server's calculated energy consumption in the cache pool. The cache pool's cache resources are limited. The edge caching model based on blockchain serves as an extension of the system cache pool. In blockchain, the processing time of tasks is represented by Eq. (12).

$$T_2=T_1+T^{find}=T_{ij}+T^{select}\left(n\right)+T^{find} \tag{12}$$

In Eq. (12), T_2 means the task processing time in the blockchain. T^{find} is the retrieval time. The terminal server's energy consumption is represented by Eq. (13).

$$E_2=E_1+e_{-1}^l T^{find}=E_{up}+e_{-1}^l T^{select}(n)+e_{-1}^l T^{find} \tag{13}$$

In Eq. (13), E_2 represents the computational energy consumption of the terminal server in the blockchain. The system does not retrieve the required task results in both the edge cache pool and the blockchain network. Under the constraint of delay, task computation can be performed on edge servers. The task calculation time is represented by Eq. (14).

$$T_i^e=\frac{r_{wj}}{f_i^e} \tag{14}$$

In Eq. (14), T_i^e represents the processing time of edge

node i. f_i^e is the computing power of i. The task processing time is represented by Eq. (15).

$$T_3 = T_2 + T^{edge} = T_{ij} + T^{select}(n) + T^{find} + T_i^e \quad (15)$$

In Eq. (15), T_3 represents the task's total processing time. Edge nodes' energy consumption is represented by Eq. (16).

$$E_3 = E_{up} + e_{-1}^l T_3 \quad (16)$$

In Eq. (16), E_3 represents the computational energy consumption of the edge server. In the industrial Internet, the cache task scheduling strategy is implemented by considering time constraints, device computing capacity and storage capacity. The cost of blockchain rewards and task scheduling strategies constitutes the total system cost. To minimize task scheduling strategy's cost consumption, an optimal system cost can be obtained, represented by Eq. (17).

$$\begin{cases} E_{all}(x,y,z) = \sum_{i=1}^{n} \begin{Bmatrix} \{(1-x_i)E_0\} \\ +\{x_i\{y_i E_1 + (1-y_i)[z_i E_2 + (1-z_i)E_3] + E_{up}\}\} \end{Bmatrix} \\ F_{all}(x,y,z) = \varepsilon E_{all}(x,y,z) - (1-z_i) R\mu_n e^{-\eta\lambda(Sn)} \end{cases} \quad (17)$$

In Eq. (17), $E_{all}(x,y,z)$ means the system's total energy consumption. $F_{all}(x,y,z)$ is the optimal cost. y and z are both the positions of task results, and $y,z \in \{0,1\}$. ε is a conversion coefficient between energy consumption and cost. Edge computing task scheduling strategy for task caching belongs to multi-constraint optimization problem. As the tasks and cached data increase, the solution space of tasks also increases. Finding the optimal feasible solution in a vast solution space is crucial. Therefore, genetic evolutionary algorithms are used to solve for the lower cost that the optimal scheduling strategy can achieve. Fig. 4 shows the Task Scheduling Algorithm based on Genetic Optimization (TSA-GO). Firstly, the task is initialized. The fitness function is used to determine the individual superiority or inferiority. By combining selection, crossover, and mutation operations, the optimal solution of the scheduling strategy is solved, achieving low-cost control.

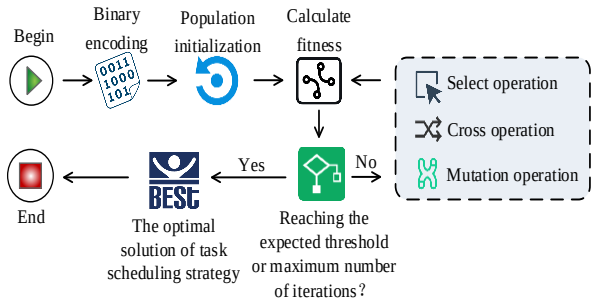


Fig. 4. Task scheduling strategy based on differential genetic evolution.

The task scheduling location adopts binary encoding, with 0 and 1 indicating that the task is executed on the local device and edge server, respectively. During initialization, TSA-GO collects task deadlines, cache status, power, and device computing power to determine task resource requirements and determine task execution locations. The fitness function is determined by blockchain rewards and task scheduling strategies, and preliminary feasible solutions that meet the conditions are obtained. This system uses roulette wheel to select individuals with high fitness and combines single-point crossover to reduce the damage of crossover to the predetermined population. To prevent ineffective mutations in the algorithm, this study evaluates the mutated nodes. Random tasks in the task sequence serve as mutation points, and the probability of individual mutation is used to determine whether the task is mutated. The demand for resources in a task serves as a mutation point, and the mutation threshold is used to determine whether to mutate.

III. Results

This experiment verified the edge computing task scheduling algorithm based on blockchain and task cache. Firstly, an analysis was conducted on the impact of task cache pools and blockchain on costs. Subsequently, TC-ILFU was compared and analyzed with other caching algorithms. Finally, the performance of TSA-GO was analyzed under different task scheduling strategies and compared with other optimization algorithms.

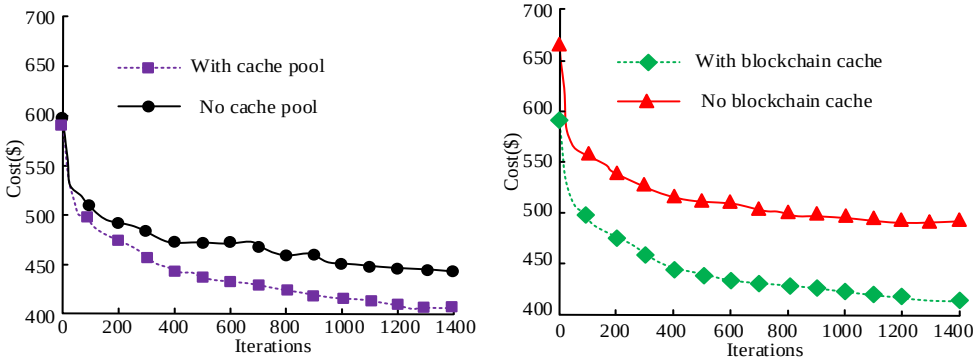
A. Experimental and Analysis of Edge Caching Model Based on Blockchain

To test blockchain based TC-ILFU, this experiment compared TC-ILFU with LFU, Least Recently Used (LRU) algorithm, and First in First Out (FIFO) algorithm. TC-ILFU was analyzed in terms of cost and hit rate. Table I shows the experimental parameters.

In Table I, the computing power of local devices was the weakest, while the computing power of edge servers increased but did not exceed that of cloud computing servers. In Fig. 5, this experiment compared the effects of task cache pool and blockchain on system costs with and without cache pool and blockchain cache.

TABLE I. Experimental Parameter Settings

Parameter	Value
Computing frequency of Local device	1GHz
Idle power of local devices	0.3W
On-load power of local devices	0.9W
Computing frequency of edge servers	4GHz
Computing frequency of cloud servers	12GHz
Number of tasks	[100, 200]
Task scale	[20, 40] Mbit
Maximum tolerance time for tasks	[0.1, 5] s
Wireless channel bandwidth	18MHz
Population size	100
Channel noise	-100dpm
Iterations	1400

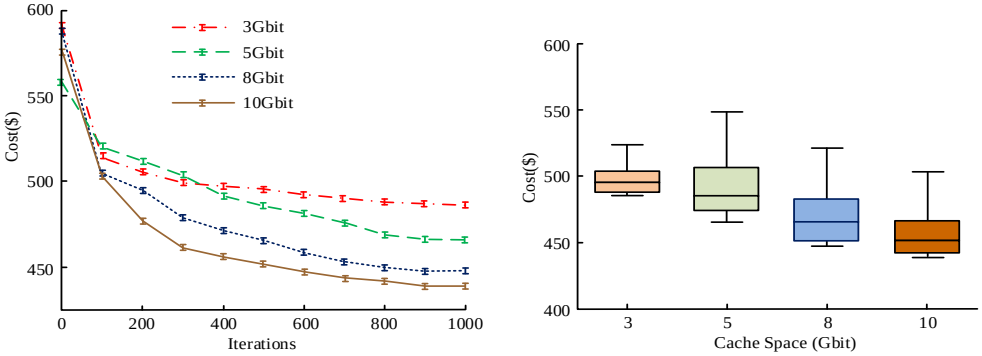


(a) The impact of cache pool on costs (b) The impact of blockchain on costs

Fig. 5. The impact of cache pool and blockchain on costs.

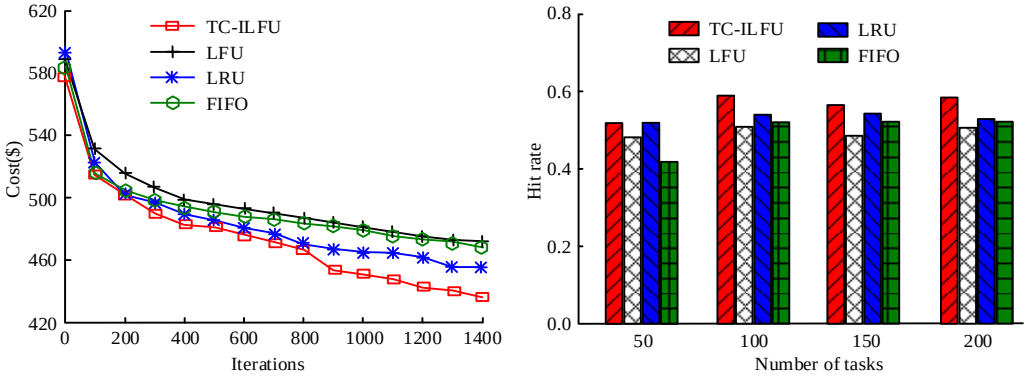
In Fig. 5(a), as the iteration increased, the cost gradually decreased. The average cost of a task cache pool was 435$, and the average cost of a non-cache pool was 480$. Compared to the situation without a cache pool, the cost with a cache pool was reduced by 9.4%. This is because the system only has the cost of task upload and edge computing, and the task cache pool can effectively reduce the cost of edge computing. In Fig. 5 (b), the average cost with blockchain caching was 450$, and the average cost without blockchain caching was 525$. The cost of having no blockchain cache was about 1.2 times that of having blockchain cache. This is because blockchain can backup all cached results, ensuring data consistency and immutability, improving the reliability and security of system data. Therefore, the introduction of cache pool and blockchain in edge computing can effectively reduce costs and save the cost of industrial Internet. Fig. 6 shows the impact of cache pool capacity space on cost under the same workload.

In Fig. 6(a), when the capacity space was 3GBit, 5Gbit, 8Gbit, and 10Gbit, the average cost of the system was 508$, 495$, 483$, and 454$. Compared to the capacity spaces of 3GB, 5Gbit, and 8Gbit, the cost of 10Gbit capacity space was reduced by 10.6%, 8.3%, and 6.0%. The larger the capacity space, the lower the system cost. In Fig. 6(b), as the capacity space of the cache pool increased, the system cost gradually decreased. This is because the cost distribution varies depending on the type of task. This experiment tested the impact of TC-ILFU on cost and hit rate in a fixed cache pool, comparing TC-ILFU with LFU, LRU, and FIFO. Fig. 7 shows the comparison results of cost and hit rate for different caching algorithms.



(a) Results of four cache spaces on cost (b) Distribution of four cache space on cost

Fig. 6. The impact of cache pool capacity space on cost.



(a) Cost of four caching algorithms (b) Hit rate of four caching algorithms

Fig. 7. Cost and hit rate of four caching algorithms.

In Fig. 7(a), as the iteration increased, the cost of all four caching algorithms decreased. The average cost of TC-ILFU, LFU, LRU, and FIFO was 478$, 500$, 489$, and 497$. Compared to LFU, LRU, and FIFO, TC-ILFU reduced costs by 4.4%, 2.2%, and 3.8%. The time task frequency table avoided the impact of frequently accessed data in the past on the current cache pool data and improved the task hit rate. In Fig. 7(b), the average hit rate of TC-ILFU, LFU, LRU, and FIFO was 0.56, 0.49, 0.53, and 0.50. Compared with LFU, LRU, and FIFO, TC-ILFU improved hit rate by 14.3%, 5.7%, and 12.0%. When the task was 50, the hit rate of TC-ILFU and LFU was consistent. This is because the repetition rate of task data is low in a relatively short period of time. As the repetitive tasks increase, the hit rate of TC-ILFU gradually increases.

B. *Experiment and Analysis of Edge Computing Task Scheduling Algorithm*

This experiment compared all local task scheduling strategies (All-local), all edge task scheduling strategies (All-edge), GA, Simulated Annealing (SA), and Hill Climbing (HC) to analyze TSA-GO from the perspectives of system latency, energy consumption, cost, and runtime. Fig. 8 shows the time and energy consumption of the system under different task scheduling strategies.

From Fig. 8(a), the average latency of All-local, All-edge, and TSA-GO was 600ms, 408ms, and 332ms. All-local had the highest latency, with all tasks being executed on local devices, resulting in task loss due to limited resources and increasing task processing time. TSA-GO effectively reduced latency by jointly processing tasks with local devices and edge servers. In Fig. 8(b), the average energy consumption of All-local, All-edge, and TSA-GO was 572J, 526J, and 272J. Compared with All-local and All-edge, TSA-GO reduced energy consumption by 52.4% and 48.3%, respectively. TSA-GO communicated resources through task cache pools and blockchain, effectively reducing the additional consumption caused by task growth. Fig. 9 shows the cost and time comparison of different algorithms.

In Fig. 9(a), as the iteration increased, the costs of all four algorithms decreased. The average cost of TSA-GO, GA, SA, and HC was 443$, 472$, 506$, and 508$, respectively. Compared with GA, SA, and HC, the cost of TSA-GO decreased by 6.1%, 12.5%, and 12.8%. This is because GA randomly generates a large number of solutions, increasing the solution space and making it difficult to find the optimal solution. SA belongs to completely greedy algorithms, and each time the current optimal solution is selected, only local optimal solutions can be searched. HC belongs to simple greedy algorithms, which select an optimal solution in the nearby solution space as the current solution until reaching a local optimal solution. TSA-GO restricts task initialization and reduces algorithm optimization time. Restricting mutation operations to avoid useless mutations can improve convergence rate. In Fig. 9(b), the running time of TSA-GO, GA, SA, and HC was 25600ms, 28880ms, 46680ms, and 35000ms, respectively. SA had the longest running time. The running time of TSA-GO and GA was moderate. TSA-GO searched for the optimal solution faster. This is because the optimization time of the algorithm is reduced when initializing the population. To further validate TSA-GO, experiments were conducted on factors such as the computing power of edge servers and the transmission rate of channels that affect system costs. Fig. 10 shows the results of the impact of different computing power on costs.

In Fig. 10(a), with the increase of edge computing frequency, the system cost gradually decreased. When the calculation frequency of edge nodes was 8GHz, 10GHz, 12GHz, 15GHz, and 18GHz, the average system cost was $330, $288, $245, $231, and $210, respectively. Compared to edge nodes with a computing frequency of 8GHz, the cost of nodes with an 18GHz frequency was reduced by 36.4%. In Fig. 10 (b), the cost reduction rate in the channel rates of 8-10GHz, 10-12GHz, 12-15GHz, and 15-18GHz range was 0.14, 0.19, 0.07, and 0.08, respectively. Cost convergence did not change regularly with the increase of edge computing frequency, and the cost reduction rate remained within 20%. Fig. 10 shows the impact of different channel rates on system costs.

In Fig. 11(a), as the channel rate increased, the tasks uploaded to the edge server increased, and cost control became more optimized. When the channel upload rate was 5M/s, 8M/s, 10M/s, 12M/s, and 15M/s, the average system cost was $465, $338, $302, $270, and $250, respectively. In Fig. 11(b), the cost reduction rate was 0.31, 0.13, 0.09, and 0.10 in the channel rates of 5-8M/s, 8-10M/s, 10-12M/s, and 12-15M/s, respectively. As the channel rate increased, the cost reduction rate gradually stabilized and remained around 10%.

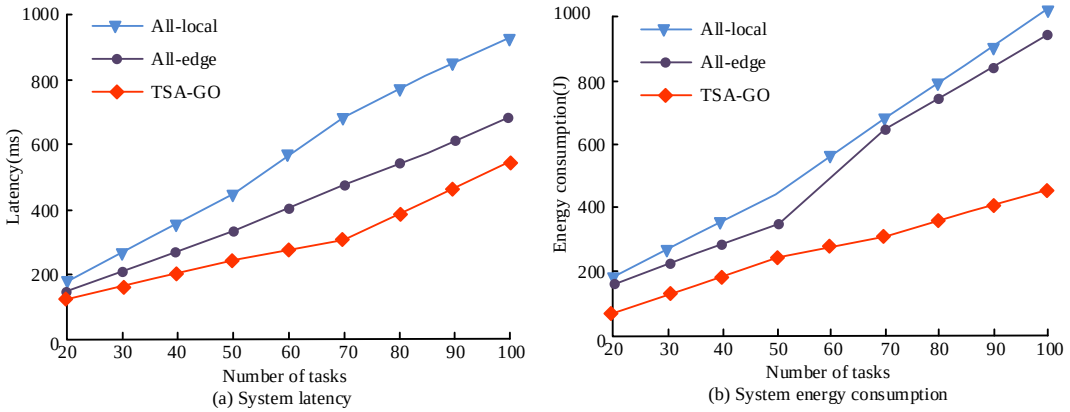


Fig. 8. Cost and running time of four algorithms.

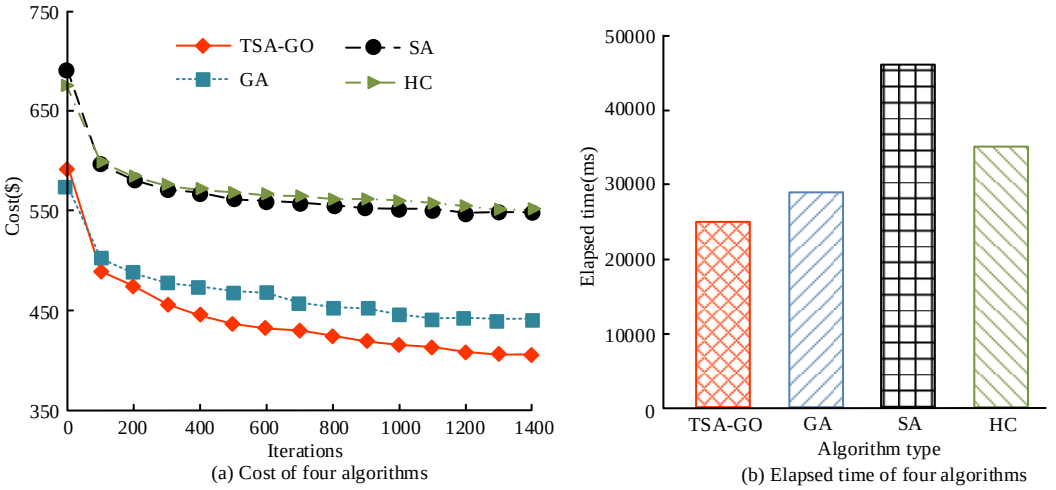


Fig. 9. Cost and elapsed time of four algorithms.

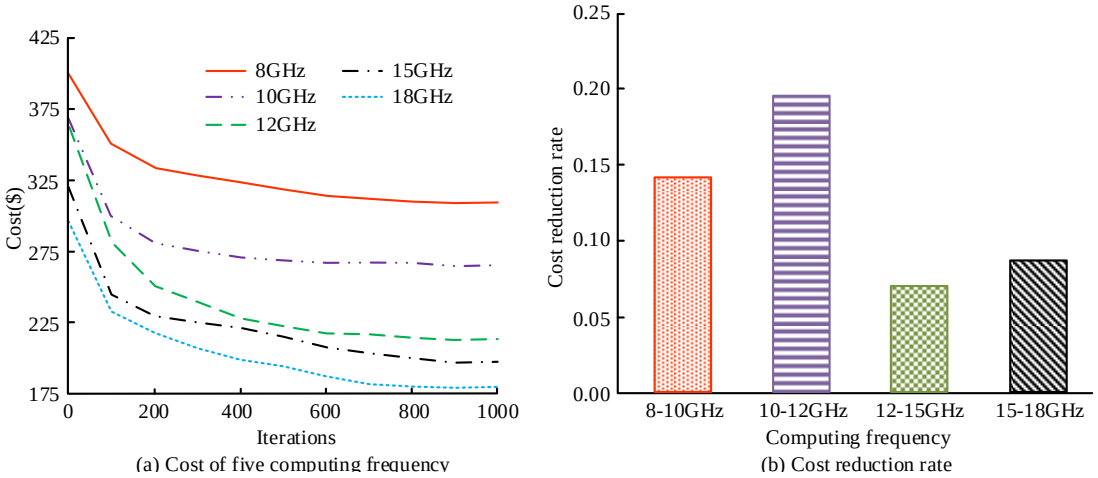


Fig. 10. Cost of five computing power.

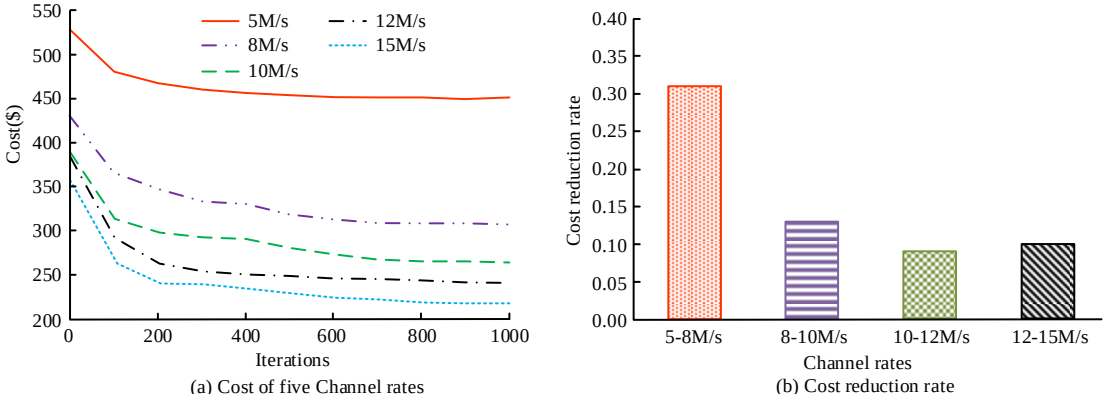


Fig. 11. Cost of five channel rates.

IV. Discussion and Conclusion

A. Discussion

To solve the transmission bandwidth limitation and data privacy threat existing in the industrial Internet, the research proposed to schedule the task to the edge server for processing to realize the real-time and security of data in industrial production. However, traditional task scheduling algorithms do not fully utilize edge cache resources, which can easily lead to the leakage of private data. Therefore, this research proposed a security enhanced edge computing task scheduling method based on blockchain and task cache. Firstly, the edge caching model of blockchain was utilized to cache the calculation results of multiple repetitive tasks in the intelligent factory,

reducing task latency. The task scheduling of coupling task caching and blockchain assisted caching was modeled as a cost minimization problem under multiple constraints. Meanwhile, the genetic optimization algorithm was combined to achieve optimal cost control. Yasir M and other researchers proposed a content caching strategy based on mobile edge computing, which significantly improved the cache performance of edge servers and increased the cache hit ratio [22]. The experimental results of this study showed that the cost of the TC-ILFU algorithm was reduced by 4.4%, and the hit rate was increased by 14.3%, which is similar to the results of Yasir M and other researchers, further confirming that the improved LFU algorithm can effectively improve the cache hit rate. Yin Z's research team has developed a multi-objective task scheduling strategy for intelligent production lines, which has a high task completion rate and can effectively reduce task service delays and energy consumption [23]. The experimental results of this study show that the TSA-GO algorithm reduces latency and energy consumption by 44.7% and 52.4%, respectively, which is consistent with the results of Yin Z's research team. The main reason is that the cloud edge end mode used in industrial task scheduling can effectively reduce cloud task processing overhead and transmission delay. Scholars such as Fu X have improved the overall completion time and convergence accuracy of cloud tasks using a hybrid particle swarm optimization genetic task scheduling algorithm [24]. This study shows that the TSA-GO algorithm reduces the cost by 6.1% and improves the system running time by about 10%, which is different from the research results of scholars such as Fu X. This is because scheduling tasks to edge services can effectively reduce cloud computing costs and accelerate task processing speed.

B. Conclusion

In conclusion, the research proposes that the security-enhanced edge computing task scheduling method based on blockchain and task cache can effectively protect data privacy, reduce latency, reduce costs, and improve system security. The limitation of the research is that the dynamic scheduling scenario of time-varying resources was not fully considered. Subsequently, a Markov strategy scheduling algorithm was used to construct a dynamic model of the industrial environment. Based on the environmental resource changes, resource allocation strategies and scheduling strategies were dynamically predicted to reduce time and resource costs in industrial scenarios.

REFERENCES

[1] Li H, Li X, Liu X, Bu X, Li H, Lyu Q. Industrial internet platforms: applications in BF ironmaking. Ironmaking & Steelmaking, 2022 49(9):905-916.

[2] Dziubinski K, Bandai M. Bandwidth Efficient IoT Traffic Shaping Technique for Protecting Smart Home Privacy from Data Breaches in Wireless LAN. IEICE Transactions on communications, 2021, 104(8): 961-973.

[3] Salem R B, Aimeur E, Hage H. A Multi-Party Agent for Privacy Preference Elicitation. Artificial Intelligence and Applications, 2023, 1(2): 98-105.

[4] Mokayed H, Quan T Z, Alkhaled L, Sivakumar V. Real-time human detection and counting system using deep learning computer vision techniques. Artificial Intelligence and Applications, 2023, 1(4): 221-229.

[5] Chen M, Zhang L. Application of edge computing combined with deep learning model in the dynamic evolution of network public opinion in emergencies. Journal of supercomputing, 2023, 79(2): 1526-1543.

[6] Liu L, Zhao M, Yu M, Jan M A, Lan D, Taherkordi A. Mobility-Aware Multi-Hop Task Offloading for Autonomous Driving in Vehicular Edge Computing and Networks. IEEE transactions on intelligent transportation systems, 2023, 24(2): 2169-2182.

[7] Meneguette R, De Grande R, Ueyama J, Rocha Filho, G P, Madeira E. Vehicular Edge Computing: Architecture, Resource Management, Security, and Challenges. ACM computing surveys, 2023, 55(1): 4-50.

[8] Shi W, Wu J, Chen L, Zhang X, Wu H. Energy-efficient cooperative offloading for mobile edge computing. Wireless networks, 2023, 29(6): 2419-2435.

[9] Gao J, Kuang Z, Gao J, Zhan L. Joint Offloading Scheduling and Resource Allocation in Vehicular Edge Computing: A Two Layer Solution. IEEE Transactions on Vehicular Technology, 2023, 72(3): 3999-4009.

[10] Chen Y, He S, Jin X, Jin X, Wang Z, Wang F, Chen L. Resource utilization and cost optimization oriented container placement for edge computing in industrial internet. Journal of supercomputing, 2023, 79(4): 3821-3849.

[11] Sharma P, Jindal R, Borah M D. Blockchain Technology for Cloud Storage: A Systematic Literature Review. ACM computing surveys, 2021, 53(4):89-120.

[12] Zhang Q, Zhao Z. Distributed storage scheme for encryption speech data based on blockchain and IPFS. Journal of supercomputing, 2023, 79(1): 897-926.0.

[13] Nguyen T, Thai M T. Denial-of-Service Vulnerability of Hash-Based Transaction Sharding: Attack and Countermeasure. IEEE Transactions on Computers, 2023, 72(3): 641-652.

[14] Zhang Q, Li C, Du T, Luo Y. Multi-level caching and data verification based on ethereum blockchain. Wireless networks, 2023, 29(2):713-727.

[15] Kong L, Tan J, Huang J, Chen G, Wang S, Jin X, Zeng P. Edge-computing-driven Internet of Things: A Survey. ACM computing surveys, 2023, 55(8): 1-41.

[16] Li X, Lan X, Mirzaei A, Bonab M J A. Reliability and robust resource allocation for Cache-enabled HetNets: QoS-aware mobile edge computing. Reliability Engineering & System Safety, 2022, 220(4): 108272-108287.

[17] Rivera A V, Refaey A, Hossain E. A Blockchain Framework for Secure Task Sharing in Multi-Access Edge Computing. IEEE Network: The Magazine of Computer Communications, 2021,35(3): 176-183.

[18] Zhang H, Wang R, Sun W, Zhao H. Mobility Management for Blockchain-based Ultra-dense Edge Computing: A Deep Reinforcement Learning Approach. IEEE Transactions on Wireless Communications, 2021, 20(11): 7346-7359.

[19] Chen J, Pu C, Wang P, Huang X, Liu Y. A blockchain-based scheme for edge–edge collaboration management in time-sensitive networking. Journal of King Saud University-Computer and Information Sciences, 2024 36(1): 101902-101918.

[20] Liu R, Yu X, Yuan Y, Ren Y. BTDSI: A blockchain-based trusted data storage mechanism for Industry 5.0. Journal of King Saud University - Computer and Information Sciences, 2023,35(8): 101674-101683.

[21] Li G, Dong Y, Li J, Song X. Strategy for dynamic blockchain construction and transmission in novel edge computing networks. Future Generation Computer Systems, 2022, 130(5): 19-32.

[22] Yasir M, uz Zaman S K, Maqsood T, Rehma, F, Mustafa S. CoPUP: Content popularity and user preferences aware content caching framework in mobile edge computing. Cluster Computing, 2023, 26(1): 267-281.

[23] Yin Z, Xu F, Li Y, Fan C, Zhang F, Han G, Bi Y. A multi-objective task scheduling strategy for intelligent production line based on cloud-fog computing. Sensors, 2022, 22(4): 1555-1575.

[24] Fu X, Sun Y, Wang H, Li H. Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm. Cluster Computing, 2023, 26(5): 2479-2488.