International Journal for Housing Science and Its Applications
Publish August 6, 2025. Volume 46, Issue 3 Pages 5882-5892

https://doi.org/10.70517/ijhsa463506

Optimization Model Design for Industrial Internet Resource Scheduling Based on Cloud-Edge Collaboration Architecture

Xingyan Shi[1,*]
[1] School of Information Engineering, Henan Vocational College of Agriculture, Zhengzhou, Henan, 451450, China
Corresponding authors: (e-mail: 13849029903@163.com).

Abstract The rapid development of industrial Internet has given rise to a large number of real-time computing demands, and traditional cloud computing cannot meet the low-latency requirements, while the cloud-edge cooperative architecture integrates the advantages of high computing power in the cloud and low latency at the edge, which has become an effective solution for resource scheduling in industrial Internet. In this paper, we design an optimization model for industrial Internet resource scheduling based on cloud-edge cooperative architecture, which realizes the cooperative processing of data and computation by integrating cloud computing and edge computing technologies. Firstly, an independent task resource scheduling model is constructed, and three evaluation indexes, namely, computing time, energy consumption and throughput rate, are defined; then a path planning method based on DDPG and a multilevel resource scheduling model at the cloud-edge end are designed to realize optimal allocation of tasks among multilevel resources. Simulation results show that the proposed CMRS model tends to converge at the 310th iteration, and the convergence speed is 30 iterations faster than GA-DDPG, and the reward value at convergence reaches -12.9, which is better than GA-DDPG's -17.1. In terms of cache hit rate, the CMRS model is significantly higher than the comparison algorithm. The performance evaluation shows that the CMRS model achieves lower total system overhead and task processing latency when dealing with tasks of different complexity and different data volumes, proving that the model is able to effectively improve resource utilization and quality of service in the industrial Internet environment.

Index Terms cloud-edge collaboration, resource scheduling, industrial internet, DDPG algorithm, multilevel scheduling, throughput rate

I. Introduction

The era of 5G Internet of Everything has arrived, and Industrial Internet, as one of the most important application scenarios of 5G, integrates information and communication technologies and advanced manufacturing industries including 5G, Internet of Things, artificial intelligence, cloud computing, etc., and forms a new type of industrial form of informationization, automation, and precision of industrial production [1], [2]. Information and communication technology has long been widely used in the field of industrial control, but all types of technology have corresponding shortcomings and defects. In most of the current industrial network environments, most of the equipment in the factory is connected through wired networks, which is not conducive to the collection and analysis of data, and it takes a lot of time and cost to deploy the network and improve the adjustment [3], [4]. Compared with wired communication technology, wireless communication technology is more intelligent and automated, has the characteristics of low cost, easy to use, flexibility and so on, can greatly improve the production efficiency, save production costs, can provide efficient communication ability in complex special industrial scenes, ensure timely and effective transmission of information and data, and make up for the shortcomings of wired networks [5]-[8].

With the continuous development of IoT technology, there will be a large number of industrial equipment (e.g., automated robots, AGVs, automatic lathes, etc.) accessed as wireless network terminal nodes. At the same time, with the industrial Internet low latency, high reliability, massive data, massive access, etc. put forward more demanding and personalized requirements for communication networks. For example, industrial automation control scenarios require real-time monitoring of numerous production components and links to detect product quality, and real-time adjustment and improvement of the assembly line, so the latency should be controlled at 10-100 ms and the reliability is guaranteed to be above 99.999% [9]-[11]. Considering that the bandwidth resources of wireless communication are limited, while the types and numbers of terminals in the industrial Internet are increasing, and the business volume of the network in the industrial Internet is not uniformly distributed, the channel characteristics fluctuate due to channel fading and interference, and the different types of industrial Internet terminals have different requirements for the performance indexes of the communication, so wireless resource management and flexible scheduling are very important [12], [13]. Designing a reasonable and efficient resource scheduling mechanism to flexibly and dynamically manage communication resources in a wireless network can improve spectrum utilization, reduce co-channel interference and enhance system throughput, reduce transmission delay, and thus prevent network congestion and reduce network signaling load [14]-[17]. In industrial Internet scenarios, industrial intelligent services have more stringent requirements for network communication metrics, such as lower latency and higher transmission reliability, etc. However, the mainstream resource scheduling schemes are inflexible, and it is difficult to balance between local optimization and global optimization, and between global optimization and throughput [18]-[20]. There is an urgent need to optimize the current resource scheduling methods to achieve stable operation of industrial Internet.

Industrial Internet, as a product of the deep integration of new-generation information technology and manufacturing, is promoting the transformation of the traditional industrial system to digitalization, networking and intelligence. With the explosive growth in the number of industrial Internet devices, massive data and large-scale computing demands are generated, posing unprecedented challenges to resource scheduling capabilities. Traditional resource scheduling methods mainly rely on cloud computing. Although cloud computing has powerful computing and storage capabilities, the centralized architecture leads to network congestion and high transmission latency, making it difficult to meet the strict real-time requirements of industrial applications. Edge computing can effectively reduce latency by sinking computing power to the edge of the network, but its limited computing and storage resources are difficult to support complex task processing. Therefore, the Cloud-Edge Collaboration Architecture, which complements the advantages of cloud computing and edge computing, has emerged as an effective way to solve the resource scheduling problem of industrial Internet. Through the four dimensions of resource synergy, data synergy, service synergy and management synergy, Cloud-Edge Collaboration Architecture realizes the organic combination of high-performance computing on the cloud side and low-latency processing on the edge side, and provides more intelligent, efficient, safe and reliable services for the Industrial Internet. However, how to realize efficient scheduling of cloud-edge resources in the complex and changing industrial Internet environment still faces many challenges. First, industrial Internet tasks are diverse and heterogeneous, requiring comprehensive consideration of multi-dimensional evaluation indexes such as computation time, energy consumption, and throughput rate; second, in the face of dynamically changing network environments and computational loads, traditional static scheduling strategies are difficult to adapt to; and lastly, the multilevel resource architectures formed by cloud, edge, and terminal computing increase the complexity of scheduling decisions. Based on this, this paper constructs an industrial Internet resource scheduling optimization model for cloud-edge collaborative architecture, which first defines an independent task resource scheduling model containing computation time, energy consumption, and throughput rate, and provides a multi-dimensional evaluation system for cloud-edge resource scheduling; and then a path planning method based on Deep Deterministic Policy Gradient (DDPG) is designed to solve the problem of task routing in a complex network topology; Finally, a multilevel resource scheduling model for cloud-side-end is proposed to achieve the cooperative optimization of computation offloading and path planning. Through comparison experiments with multiple benchmark solutions, the proposed model is verified to be superior in terms of model convergence speed, cache hit rate, and task processing overhead, which provides a new solution for resource scheduling optimization in the industrial Internet.

II. Resource scheduling optimization model construction under cloud-side collaboration

II. A. Cloud Edge Collaboration

Cloud computing and edge computing are two different computing models. Cloud computing provides computing, storage and networking services by storing data and applications on remote servers and is good at handling large amounts of data and performing complex computing tasks. Edge computing, on the other hand, performs computation and storage in close proximity to the data source, aiming to reduce data transmission and processing time, improve response time and data security, and is suitable for processing real-time data and application scenarios that require fast response. Cloud-edge collaboration is a new type of collaborative cooperation model [21], [22], which combines cloud computing and edge computing technologies to achieve collaborative processing of data and computation. In this model, cloud and edge can collaborate with each other and utilize their respective advantages to achieve higher efficiency and better user experience. The core idea of cloud-edge collaboration is to distribute data and computation tasks to the cloud and edge for processing. The cloud provides powerful computing and storage capabilities, capable of undertaking large-scale data processing and analysis, as well as providing services on a global scale; while the edge provides more proximity and fast computing and storage capabilities, enabling real-time data processing and response. Through cloud-edge collaboration, more intelligent, efficient, safe and reliable services can be realized. The main meanings of cloud-side collaboration include resource collaboration, data collaboration, service collaboration and management collaboration:

(1) Resource synergy: Resource synergy of cloud computing and edge computing refers to the close synergy of computing and storage resources of both to realize more efficient computing and storage. Cloud computing has large-scale, high-performance, powerful computing and storage capabilities, which can be used to perform complex data analysis and processing; while edge computing is closer to the user and can provide faster data processing and response speed. Therefore, in cloud-edge collaboration, cloud computing and edge computing can share each other's resources to achieve more efficient computing and storage.

(2) Data Collaboration: data collaboration refers to data sharing and exchange between cloud computing and edge computing. In cloud-edge synergy, cloud computing can provide large-scale and complex data analysis and processing, while edge computing can provide real-time data collection and preprocessing. Therefore, in data synergy, cloud computing and edge computing can share each other's data to achieve more efficient data processing and analysis.

(3) Service Collaboration: service collaboration refers to the sharing and exchange of services between cloud computing and edge computing. In cloud-edge synergy, cloud computing can provide large-scale and complex services and applications, while edge computing can provide real-time services and applications. Therefore, in service collaboration, cloud computing and edge computing can share each other's services and applications to achieve more efficient service and application delivery.

(4) Management Collaboration: management collaboration between edge devices and cloud. The cloud can remotely manage and monitor the status and operation of edge devices, send down configuration information and tasks, and diagnose and troubleshoot device failures. This can improve the management efficiency and operational stability of the equipment, and also reduce the load pressure on the cloud and improve the overall computing and storage resource utilization.

II. B.Independent Task Resource Scheduling Model

The cloud edge collaboration architecture consists of endpoints, offload agents and cloud edge resource pools. The cloud edge resource pool is further subdivided into edge resource pool and cloud resource pool. The edge resource pool consists of multiple edge nodes while the cloud resource pool consists of multiple cloud nodes. The offloading agent is an intelligent program module that can be deployed at the edge gateway, where the endpoints contain a variety of smart devices such as smart watches and surveillance cameras, which are installed with a wide variety of applications and make different processing requests to the cloud edge resources. The workflow is as follows: first, the agent accepts requests from the endpoints and decomposes them into multiple independent tasks; subsequently, the agent obtains the data volume and the maximum tolerable processing time for each task and finds the optimal processing node for each task by running a scheduling algorithm based on the current node state and task attributes. This process ensures that tasks can be quickly and efficiently offloaded to appropriate processing nodes, thus realizing efficient collaboration of cloud-side resources.

Define $T = \{T_1, T_2, ..., T_n\}$ denotes a set of tasks, and the set of features of the tasks is $T_i = \{T_i^{da}, T_i^{de}\}$, where T_i^{da} is the amount of data carried by the i th task, T_i^{de} is the amount of data the i th task can tolerate for the maximum processing time. The cloud edge resource pool consists of a set of cloud edge nodes $N = \{N_1, N_2, ..., N_m\}$, each node can be represented by a quaternion $N_j = \{N_j^r, N_j^d, N_j^w, N_j^{id}\}$, where N_j^r denotes the CPU processing rate of the first j node and N_j^d denotes the interaction delay between the jth node and the agent when offloading the task, and N_j^w and N_j^{id} denote the working power and idle power of the jth node, respectively.

II. B. 1) Computational time modeling

Define x_{ij} to denote the decision result of whether the i th task is offloaded to the jth node or not, as shown in Equation (1):

$$x_{ij} = \begin{cases} 1 & \text{if the } i\text{th task computed by the } j\text{th node} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

An offloading strategy matrix x_{ij} is formed from $x_{(n\times m)} = \{x_{ij} \mid 1 \le i \le n, 1 \le j \le m\}$. In order to utilize the computational resources more efficiently, each task is computed by only a single computational node, see Equation (2):

$$\sum_{j=1}^{m} x_{ij} = 1, i \in \{1, 2, ..., n\} \tag{2}$$

In order to obtain the total computation time, it is also necessary to obtain the computation time N_j at the processing node t_{ij} for each task T:

$$t_{ij}=\frac{T_i^{da}}{N_j^r} \tag{3}$$

Thus, the total computation time t^{total} can be expressed as follows:

$$t^{total}=\sum_{i=1}^{n}\sum_{j=1}^{m}t_{ij}\times x_{ij} \tag{4}$$

II. B. 2) Energy consumption models

The total energy consumption of the system consists of two parts, the working energy consumption generated by the nodes when they are processing the tasks and the energy consumption generated by the nodes when they are in idle state. The working time of a node can be calculated according to the following equation:

$$t_j^w=\sum_{i=1}^{n}t_{ij}\times x_{ij} \tag{5}$$

Before defining the idle time of a node, the longest completion time M is defined first. M is the time when all tasks are fully completed, because multiple nodes process tasks in parallel, so M can be expressed as the longest working time of a node, as shown in Equation (6):

$$M=\max_{j\in\{1,2,\ldots,m\}}(t_j^w) \tag{6}$$

According to Eq. (5) and Eq. (6), the idle time of node N_j can be expressed as:

$$t_j^{id}=M-t_j^w \tag{7}$$

Based on Eq. (5) and Eq. (7), the energy consumption of node N_j is defined as follows:

$$e_j=N_j^w\times t_j^w+N_j^{id}\times t_j^{id} \tag{8}$$

The total energy consumption of the system is the sum of operating and idle energy consumption and is calculated using the following formula:

$$e^{total}=\sum_{j=1}^{m}e_j \tag{9}$$

II. B. 3) Throughput rate model

This subsection defines the response time T_i of the task r_i, which consists of the execution time of the task T_i and the interaction latency between the agent and the processing node, as shown in Eq. (10):

$$r_i=\sum_{j=1}^{m}(t_{ij}+2\times N_j^d)\times x_{ij} \tag{10}$$

Next, S is defined to denote the set of tasks that satisfy the latency requirement, and when $r_i\le T_i^{de}$ indicates that the task T_i can be processed on time in accordance with the scheduling scheme, the task is included in the set S. In addition to the two most common evaluation metrics, energy consumption and computation time, this subsection defines a new evaluation metric, throughput rate.

Throughput rate is one of the key metrics for measuring the effectiveness of a scheduling scheme [23], and is used to evaluate the number of tasks that a scheduling scheme is able to complete on time for a given maximum processing time requirement. Specifically, the throughput rate reflects how many tasks are successfully processed within the specified time, thus visually assessing the strengths and weaknesses of the scheduling scheme. A high throughput rate indicates that the scheduling solution performs well in meeting the task processing time requirements, which helps to improve the overall system performance and user satisfaction. Therefore, by monitoring and analyzing the throughput rate, it can provide a strong basis for optimizing the scheduling strategy and ensure that more tasks are completed on time. In summary, the throughput rate R is defined as the ratio of the number of tasks completed within the maximum delay to the total number of tasks, and is calculated as shown in Equation (11):

$$R=\frac{|S|}{n} \tag{11}$$

where n indicates the total number of tasks.

II. C.Multi-level resource scheduling model for cloud-side endpoints

II. C. 1) Design of DDPG-based path planning methods

The path planning problem can be described as minimizing the system transmission cost:

$$\min\sum_{t=1}^{\tau}O_t \tag{12}$$

Considering weighted routing, we design a global weighted intelligent path planning method (DDPG-WR) based on the DDPG deep reinforcement learning algorithm [24], which is suitable for continuous action space, because the complexity of the network topology, the link state space, and the weighted action space are continuous and high-dimensional, and the network state is in constant dynamic change.

State space S: In our problem since the intelligences do global routing, the state $s_t \in S$ at time frame t consists of the normalized link residual bandwidth, $|s_t| = E$, denoted by

$$s_t = \left[B_{1,t}, B_{2,t}, \ldots, B_{E-1,t}, B_{E,t}\right] \tag{13}$$

Action space A : the Actor network of DDPG intelligences assigns weights to each link, and the action $a_t \in A, |a_t| = E$ at time frame t, is denoted as:

$$a_t = [w_{1,t}, w_{2,t}, \ldots, w_{E-1,t}, w_{E,t}] \tag{14}$$

Immediately after that, the weight of each path P_z in the set of candidate paths P can be calculated based on the link weights output by the action network:

$$W_Z = \sum_{l_k \in P_z, P_z \in \mathrm{P}} w_{k,t} \tag{15}$$

Thus, the optimal path P_t^* for task transmission is the path corresponding to the minimum weight:

$$\arg\min_{P_z} W_z \tag{16}$$

Reward function: after transmitting the task according to the path planned by the intelligence for it, the intelligence will receive an immediate reward from the environment, which should reflect the optimization goal, and for the purpose of unification of magnitude, the reward function is designed as:

$$r_t = -(w_T Tanh(T_t^{net} / 100) + w_B LB_t^{net}) - \zeta \tag{17}$$

where ζ is a penalty term, where ζ is a constant real number greater than 0 if there are links with 0 remaining bandwidth resources on the transmission path planned for the task, and $\zeta = 0$ if all links through which the transmission path passes have sufficient remaining bandwidth resources.

Thus the goal of DDPG is to find the strategy μ that can maximize the cumulative return:

$$G_t = \sum_{m=t}^{T} \gamma^{m-t} r_t \tag{18}$$

The DDPG contains a total of four networks, namely the online Actor network μ, the online Critic network Q, the target Actor network μ', the target Critic network Q', and the network parameters are denoted as $\theta^{\mu}, \theta^{Q}, \theta^{\mu'}, \theta^{Q'}$. At each time step t, the online Actor network is responsible for directly outputting the deterministic policy $\mu(s_t)$ based on the input link states s_t, and in order to equip the intelligences with the ability to explore, the noise N_t is added to the output of the Actor network, so that the weight vector of the final output network links is $a_t = \mu(s_t) + N_t$, and then the optimal path is computed for scheduling according to Eq. (16), and the intelligent body will get the reward r_t from the environment and move to the next state s_{t+1} by transferring the interaction information (s_t, a_t, r_t, s_{t+1}) is deposited into the experience pool.

When the data in the experience pool reaches a certain number, a batch of experiences (s_i, a_i, r_i, s_{t+1}) is randomly sampled from it and played back for updating the online network, which is updated by the online Critic network by executing gradient descent on the loss function shown in Eq. (19):

$$L(\theta^Q) = E[(y_i - Q(s_i, a_i; \theta^Q))^2] \tag{19}$$

where the objective value $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}; \theta^{\mu'}); \theta^{Q'})$.

The online Actor network is updated by the sampled strategy gradient, which can be calculated by Eq. (19):

$$\nabla_{\theta^{\mu}} J(\theta^{\mu}) \approx \frac{1}{\tau} \sum_t \nabla_a Q(s, a; \theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu(s; \theta^{\mu})|_{s_i} \tag{20}$$

Use soft update for the target network parameters, τ is the soft update factor, so the target value y_t will change slowly, thus improving the stability of learning:

$$\begin{aligned} \theta^{Q'} &\leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \\ \theta^{\mu'} &\leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \end{aligned} \tag{21}$$

II. C. 2) Multi-level resource scheduling model design

The complete scheduling process of computing tasks includes selecting the best computing service node among the multilevel arithmetic resources at the cloud edge end and planning the optimal path for the tasks that need to be further transmitted in the edge network to reach the remote edge nodes, so a comprehensive arithmetic and network multilevel resource scheduling model for the cloud edge end is designed jointly with the D3QN computational offloading method and the DDPG-WR path planning method as shown in Fig. 1.

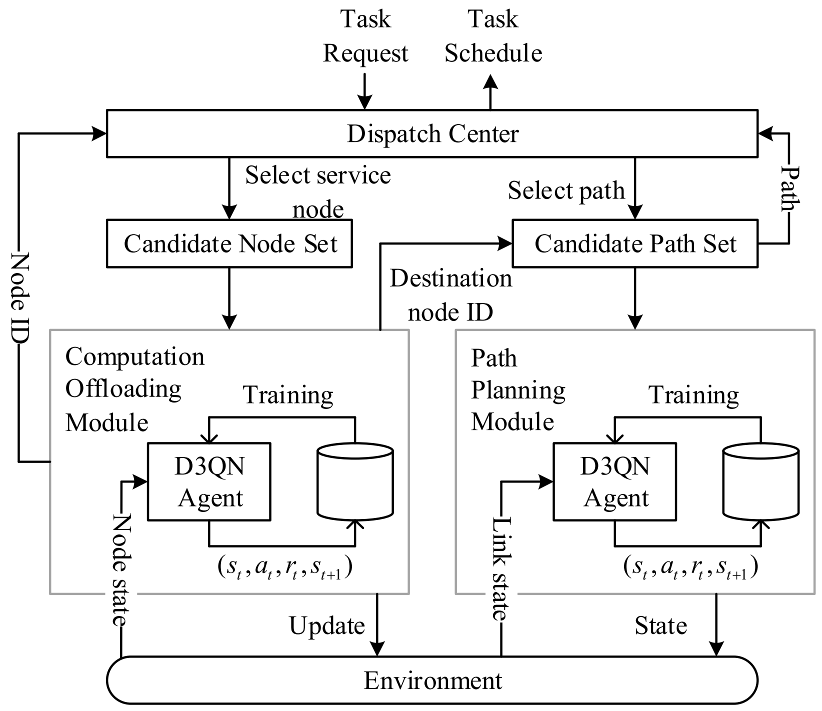

Figure 1: Multilevel resource scheduling model

The scheduling model mainly consists of two modules, the computation offload module and the path planning module. When a user requests computing services from the computing power management platform through terminal devices, the D3QN intelligences in the computation offloading module first decide the best computing node for the task based on the task request information, terminal device information, and the control plane-perceived state information of the computing power resources for the task i in the set of candidate nodes that contains three levels of computing resources, namely, cloud, edge, and end:

$$a_i^o = \arg\max_{a_i^o \in N_i} Q(s_i^o, a_i^o) \in \{0,1,\ldots,M,M+1\} \tag{22}$$

If $a_i^o = 0$ or $a_i^o = M+1$, it indicates that the decision result of computation offloading is either local computation or offloading to the cloud for execution, at this time, the task will be scheduled directly according to the decision result and the decision is finished.

And if the decision result of the computation offload module is offloaded to the edge node for computation, i.e., $a_i^o \in \{1,2,\ldots,M\}$. Remember that the routing node directly connected to the computing service node a_i^o is v_{dst}, and the edge gateway for the terminal device to access the edge layer network is v_{src}, at this time, it is necessary to further determine whether v_{src} is equal to v_{dst}, and if it is equal, it shows that the task offloads to the edge gateway which does not need to be transmitted in the edge layer network, and the task will be scheduled directly according to the decision result as well, and the decision is finished.

Only when $v_{src} \neq v_{dst}$, the task needs to be transmitted end-to-end in the edge layer network with v_{src} as the source node and v_{dst} as the destination node. So the decision result a_i^o output from the computational offloading module is used as the input to the path planning module to find the candidate path set P_i based on the corresponding source and destination nodes, and then the DDPG intelligences in the path planning module will make a global routing decision based on the network resource state information s_i^n collected from the control plane, and output the network links' weight a_i^n, and finally the optimal path P_i^* is calculated based on the weights in the set of candidate paths for scheduling, and the decision is finished.

III. Analysis of simulation results

III. A. Model convergence

In order to verify the effectiveness of the proposed algorithm, the following four benchmarking schemes are designed:

1) Genetic Algorithm Based DDPG Algorithm (GA-DDPG): the deployment of cache services is optimized using genetic algorithm, while computational offloading and resource allocation are performed using DDPG algorithm.

2) Randomized Cache Offloading Scheme (RC): the placement of cache services as well as computational offloading is done randomly.

3) DQN-based collaborative caching and offloading scheme (DQN): both service caching and computation offloading are jointly output by the DQN network, with the same parameter settings as in this method.

4) Computation offloading and resource allocation without cache (No-Cache): edge servers have no cache, which means that end devices need to offload tasks to the cloud for execution.

In this paper, the model convergence comparisons of the cloud-edge-end multilevel resource scheduling model (CMRS), GA-DDPG and DQN are shown in Fig. 2. Fig. 2 compares the performance of the convergence of the three algorithms CMRS, GA-DDPG and DQN at a learning rate of 0.001 and an iteration number of 500. Among them, CMRS and GA-DDPG present faster convergence speed with higher rewards relative to DQN, and are able to find the near-optimal policy faster. In terms of convergence speed, CMRS tends to converge at the 310th iteration, compared to GA-DDPG, which tends to converge at the 340th iteration, which indicates that CMRS converges faster than GA-DDPG, and it is able to find an optimal strategy in a shorter training time. In terms of convergence performance, CMRS converges with a reward value of -12.9 and GA-DDPG with a reward value of -17.1, which indicates that CMRS finds a better optimization strategy. From the experimental results, the model CMRS proposed in this paper outperforms the other two benchmark algorithms in both convergence speed and convergence performance. The validity and feasibility of the algorithm proposed in this paper is verified by the comparison of convergence performance.

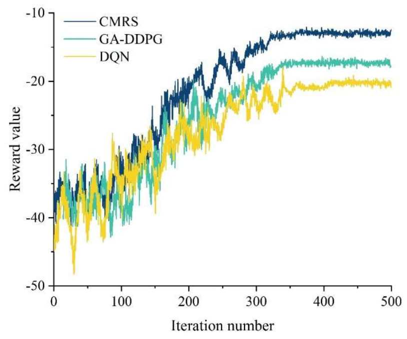

Figure 2: Model convergence comparison results

The model convergence of the CMRS model with different batch sizes is shown in Figure 3. As the number of iterations increases, the training reward value grows and stabilizes after a certain number of iterations. A smaller batch size speeds up training but may introduce more randomness. Larger batch sizes require the gradient of more samples to be computed at each update. By reducing the variance between samples, it effectively avoids oscillations and instability during training, but its training speed is slower. From the experimental results, when the batch size is 32, the training speed is faster, and it is close to the final convergence value at 240 generations, but it does not converge until 370 generations. When the batch size is 128, the training speed is slower and the algorithm falls into local optimum between 210 and 250 generations. Combining the above factors, the batch size in the CMRS model is set to 64.

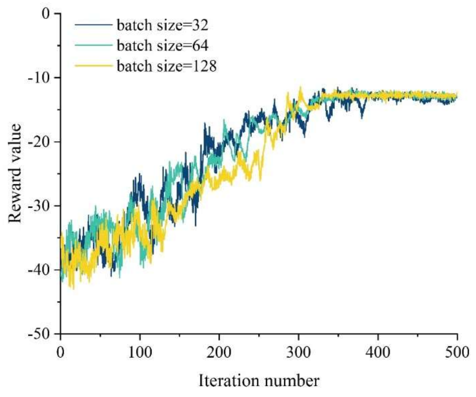

Figure 3: The model converge under different batch sizes

III. B. Cache Hit Ratio

In order to validate the performance of cache deployment optimization algorithms on large time scales, the cache hit ratio is utilized to evaluate the match-based cache deployment optimization scheme. Cache hit ratio refers to the ratio between the number of requested service cache hits and the total number of requests.

The ratio of cache hit ratio versus the number of mobile terminals under different schemes is shown in Fig. 4. Since there is no cache service in the edge server of the No-Cache algorithm, the cache hit rate is not considered since the offloaded tasks are transferred to the cloud for processing. As shown in Fig. 4, with the increasing number of terminals, the locality and repetitiveness of the computation tasks increase, which leads to the increasing cache hit rate of the edge servers.The RC algorithm has the worst cache hit rate compared to other algorithms, which is due to the randomized way of service deployment on the edge servers. In contrast, the cache hit rate of the CMRS model proposed in this paper outperforms that of GA-DDPG and DQN, and this advantage mainly stems from the fact that the cache deployment problem is converted into a one-to-many matching problem in terms of the deployment method of the services by creating preference lists for the services and the edge servers, respectively, which leads to an increase in the cache hit rate of the services.

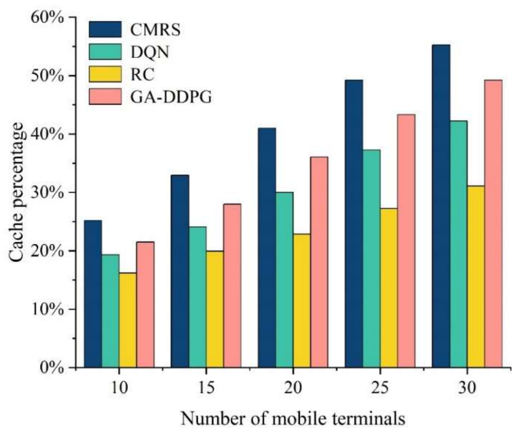

Figure 4: The comparison results of the cache hit ratio and the mobile terminal in different schemes

III. C. Analysis of equipment overhead

The comparison curves of the total system overhead of the proposed cloud-side-end multilevel resource scheduling model (CMRS) with five existing architectures (local processing architecture, WiFi-Cloud, BS-Cloud, WiFi-MEC, and BS-MEC) based on the amount of task data are shown in Fig. 5. As the amount of task data increases, the total overhead corresponding to both the architecture proposed in this paper and the four offloading strategy architectures increases, while the total system overhead of the local processing architecture remains stable, and the offloading strategy under the architecture proposed in this paper is the most effective as can be seen from the total system overhead curve. The reason is that the offloading strategy based on Q-learning training iterations fully considers the computational tasks and environmental conditions during the offloading process, and reduces the unnecessary system loss due to network congestion and node anomalies.

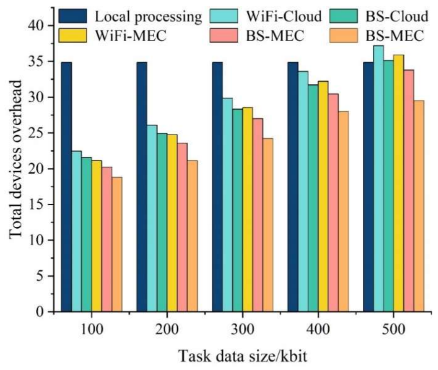

Figure 5: Relationship between total devices overhead and task data size

The comparison curves of the total system overhead based on task complexity of the proposed cloud-edge-end multilevel resource scheduling model (CMRS) with five existing architectures of five existing architectures are shown in Fig. 6. As the task complexity rises, the corresponding total system overhead gradually rises for all architectures. However, the offloading strategy under the CMRS model architecture in this paper can more reasonably cope with computational tasks of different complexity by favoring local processing when the task is relatively simple and offloading processing when it is relatively complex, which effectively utilizes the computational resources of the server. Therefore, compared with the existing architecture, the total system overhead under the proposed cloud-side-end multi-level resource scheduling model in this paper is lower year-on-year.

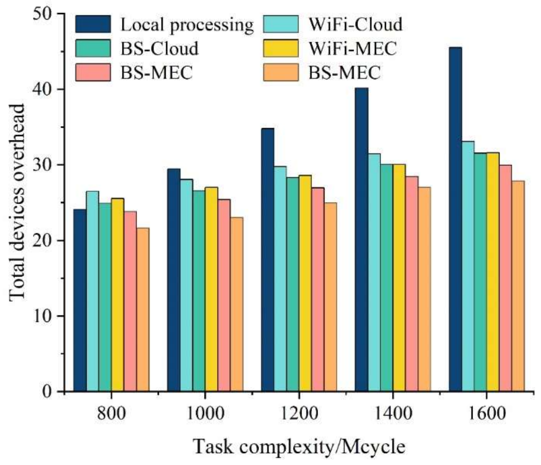

Figure 6: Relationship between total devices overhead and task complexity

The comparison curves of the task processing latency and overhead of the proposed cloud-side-end multilevel resource scheduling (CMRS) model in this paper and the five existing architectures based on the amount of task data are shown in Fig. 7 and Fig. 8, respectively. As shown in Fig. 7 and Fig. 8, the task processing latency of the local CMRS model remains stable due to the constant task complexity, and the processing overhead is always 0. In contrast, the task processing latency and overhead of the CMRS model in this paper and the four offloading architectures correspond to the task processing latency and overhead that gradually increase with the increase of the amount of task data, but with varying improvement trends. The reason is that different offloading architectures have their own performance characteristics, e.g., Wi Fi-MEC and Wi Fi-Cloud architectures do not have enough advantages in task processing latency, but can save more processing overhead. Therefore, it is necessary to formulate reasonable computation offloading strategies based on the task attributes of different computation tasks, as well as the network and server states. The CMRS model proposed in this paper synthesizes the advantages of the five existing architectures and realizes real-time decision making based on DDPG, which is more adaptable to the changing industrial Internet system environment and saves more non-essential resource loss by intelligently scheduling the processing of computing tasks.

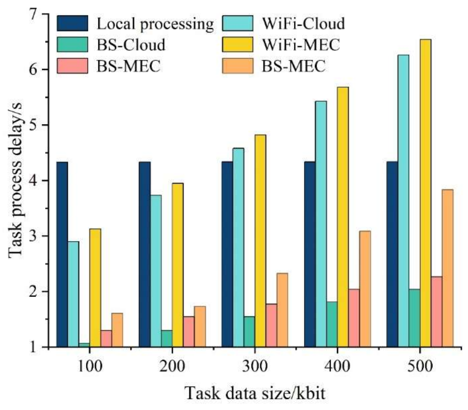

Figure 7: Relationship between task process delay and data size

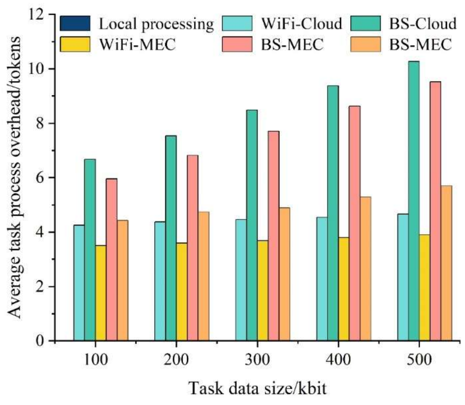

Figure 8: Relationship between task process overhead and data size

IV. Conclusion

In this paper, we designed an industrial Internet resource scheduling optimization model based on the cloud-side collaboration architecture, and realized the efficient collaboration of cloud-side resources by constructing an independent task resource scheduling model and a multilevel resource scheduling model on the cloud-side end.

Simulation results show that the proposed CMRS model outperforms the comparison algorithms in terms of convergence speed and performance, and achieves convergence at the 310th iteration, which is 30 iterations earlier than GA-DDPG, and the convergence reward value reaches -12.9, which is higher than the -17.1 of GA-DDPG and the lower value of DQN. In the cache hit rate evaluation, as the number of terminals increases, the cache hit rate of the CMRS model is significantly higher than that of the random cache offloading scheme. The total system overhead evaluation shows that the CMRS model consistently maintains the lowest overhead as the amount of task data and task complexity increases. Compared to the five existing architectures, the CMRS model performs well in terms of task processing latency, which is especially suitable for processing large data volume tasks. In addition, the CMRS model realizes real-time decision making through DDPG, which can dynamically adapt to the changing environment of the industrial Internet system, and achieves the optimal performance balance when the batch size is 64. Overall, the model proposed in this paper realizes the intelligent scheduling of multilevel resources at the cloud-side end, and provides an effective solution to improve the resource utilization and service quality in the industrial Internet environment.

References

[1] Meira, J., Matos, G., Perdigão, A., Cação, J., Resende, C., Moreira, W., ... & Aguiar, R. L. (2023). Industrial Internet of things over 5G: A practical implementation. Sensors, 23(11), 5199.

[2] Chi, H. R., Wu, C. K., Huang, N. F., Tsang, K. F., & Radwan, A. (2022). A survey of network automation for industrial internet-of-things toward industry 5.0. IEEE Transactions on Industrial Informatics, 19(2), 2065-2077.

[3] Danielis, P., Puttnies, H., Schweissguth, E., & Timmermann, D. (2018, September). Real-time capable internet technologies for wired communication in the industrial IoT-a survey. In 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA) (Vol. 1, pp. 266-273). IEEE.

[4] Underberg, L., Kays, R., Dietrich, S., & Fohler, G. (2018, May). Towards hybrid wired-wireless networks in industrial applications. In 2018 IEEE Industrial Cyber-Physical Systems (ICPS) (pp. 768-773). IEEE.

[5] Bahri, N., Saadaoui, S., Tabaa, M., Sadik, M., & Medromi, H. (2021). Wireless technologies and applications for industrial Internet of Things: A review. Advances on Smart and Soft Computing: Proceedings of ICACIn 2020, 505-516.

[6] John, J., Firyaguna, F., Sherazi, H. H. R., Kushch, S., Vijayan, A., O'Connell, E., ... & Armstrong, E. (2023). Wireless communications for smart manufacturing and industrial IoT: Existing technologies, 5G and beyond. Sensors, 23(1), 73.

[7] Huang, X. (2020). Quality of service optimization in wireless transmission of industrial Internet of Things for intelligent manufacturing. The International Journal of Advanced Manufacturing Technology, 107(3), 1007-1016.

[8] Wu, H., Lyu, X., & Tian, H. (2019). Online optimization of wireless powered mobile-edge computing for heterogeneous industrial internet of things. IEEE Internet of Things Journal, 6(6), 9880-9892.

[9] Alabadi, M., Habbal, A., & Wei, X. (2022). Industrial internet of things: Requirements, architecture, challenges, and future research directions. IEEE Access, 10, 66374-66400.

[10] Xu, H., Yu, W., Griffith, D., & Golmie, N. (2018). A survey on industrial Internet of Things: A cyber-physical systems perspective. Ieee access, 6, 78238-78259.

[11] Zhang, X., & Ming, X. (2024). A smart system of Customer-product Interaction Life Cycle (CILC) in industrial Internet era for mass personalization from industrial practice survey: identification, definition, acquisition and parsing. Journal of Intelligent Manufacturing, 35(2), 727-756.

[12] Ji, X., Tian, J., Zhang, H., Wu, D., & Li, T. (2023). Joint device selection and bandwidth allocation for cost-efficient federated learning in industrial internet of things. IEEE Internet of Things Journal, 10(10), 9148-9160.

[13] Abuhasel, K. A., & Khan, M. A. (2020). A secure industrial internet of things (IIoT) framework for resource management in smart manufacturing. IEEE Access, 8, 117354-117364.

[14] Chen, Y., Liu, Z., Zhang, Y., Wu, Y., Chen, X., & Zhao, L. (2020). Deep reinforcement learning-based dynamic resource management for mobile edge computing in industrial internet of things. IEEE Transactions on Industrial Informatics, 17(7), 4925-4934.

[15] Liang, F., Yu, W., Liu, X., Griffith, D., & Golmie, N. (2020). Toward computing resource reservation scheduling in Industrial Internet of Things. IEEE Internet of Things Journal, 8(10), 8210-8222.

[16] Wang, K., Zhou, Y., Liu, Z., Shao, Z., Luo, X., & Yang, Y. (2020). Online task scheduling and resource allocation for intelligent NOMA-based industrial Internet of Things. IEEE Journal on Selected Areas in Communications, 38(5), 803-815.

[17] Shi, Z., Xie, X., Lu, H., Yang, H., Kadoch, M., & Cheriet, M. (2020). Deep-reinforcement-learning-based spectrum resource management for industrial Internet of Things. IEEE Internet of Things Journal, 8(5), 3476-3489.

[18] Yu, P., Yang, M., Xiong, A., Ding, Y., Li, W., Qiu, X., ... & Cheriet, M. (2020). Intelligent-driven green resource allocation for industrial Internet of Things in 5G heterogeneous networks. IEEE Transactions on Industrial Informatics, 18(1), 520-530.

[19] Ojo, M. O., Giordano, S., Adami, D., & Pagano, M. (2018). Throughput maximizing and fair scheduling algorithms in industrial Internet of Things networks. IEEE Transactions on Industrial Informatics, 15(6), 3400-3410.

[20] Lin, K., Gao, J., Han, G., Wang, H., & Li, C. (2022). Intelligent blockchain-enabled adaptive collaborative resource scheduling in large-scale industrial internet of things. IEEE Transactions on Industrial Informatics, 18(12), 9196-9205.

[21] Jiasheng Chen,Miao Wang,Zhenfu Cao,Xiaolei Dong & Liwang Sun. (2025). Secure and controllable cloud–edge collaborative data sharing scheme for wireless body area networks in IIoT. Computers & Security,153,104389-104389.

[22] Xue Jiang,Haie Dou,Lei Wang & Zhijie Xia. (2025). Joint optimization of energy consumption and latency for task offloading in cloud–edge collaboration system. Physical Communication,70,102635-102635.

[23] M. Mohamed Asan Basiri. (2025). High Throughput Instruction-Data Level Parallelism Based Arithmetic Hardware Accelerator. International Journal of Parallel Programming,53(2),6-6.

[24] Fei Zhou,Lihong Zhao,Xiaomei Ding & Shuqin Wang. (2025). Enhanced DDPG algorithm for latency and energy-efficient task scheduling in MEC systems. Discover Internet of Things,5(1),40-40.