Design of Big Data Task Scheduling Optimization Algorithm Based on Improved Deep Q-Network

Fu Chen[1], Chunyi Wu[2]*
School of Smart Health, Chongqing College of Electronic Engineering, Chongqing, 401331, China[1]
Artificial Intelligence and Big Data College, Chongqing College of Electronic Engineering, Chongqing, 401331, China[2]

***Abstract*—Big data analysis can provide valuable insights not easily obtained from traditional data scales. However, addressing scheduling issues in big data can be challenging due to the vast amount and diverse nature of the data. To overcome this, a scheduling model based on Markov decision process is proposed. The deep Q-network algorithm is used for directed acyclic graph task scheduling. To improve this model further, the gradient strategy algorithm is introduced. From the results, when the dataset size was about 500, the hybrid algorithm achieved a recall rate of 0.96, outperforming the gradient strategy algorithm (0.83), deep Q-network algorithm (0.79), and estimated earliest completion time algorithm (0.63). Although the estimated earliest completion time algorithm had longer training times under different dataset sizes, the hybrid algorithm's training time was slightly longer than the gradient strategy algorithm and slightly shorter than the deep Q-network algorithm. Overall, the proposed algorithm exhibits superior performance and significant value in solving engineering problems.**

***Keywords*—Big data; Task scheduling; Policy gradient; Deep Q-network**

I. Introduction

Big data refers to a data collection generated due to its large volume, diverse types, and inability to be processed by traditional processing methods. These data typically have high speed and high diversity [1]. Compared with traditional data, big data has a larger data scale, more data types, and lower value density [2]. The data volume of big data is basically calculated at the PB level. Therefore, analyzing big data requires extremely high computational power. However, at current, the processing power of a single processor has reached its limit. Relying solely on increasing processor frequency cannot meet the current demand for big data analysis. Influenced by the development of cloud computing technology, more enterprises and research institutions are inclined to use big data analysis platforms to complete data analysis work. Traditional task scheduling algorithms are usually based on static rules, which may be inflexible and unable to adapt to real-time changing environments. In the big data environment, the nature of tasks and the availability of resources may dynamically change, which makes traditional algorithms unable to effectively cope. Some traditional algorithms may become complex when processing large-scale data, leading to an increase in computational complexity. Meanwhile, it is easy to fall into local optima, which can affect the performance of task scheduling. Therefore, a scheduling model based on Markov decision process is proposed. This model applies the Deep Q-network (DQN) algorithm to task scheduling in Directed Acyclic Graph (DAG). Then, to address the shortcomings of the DQN algorithm, a Policy Gradient (PG) algorithm is introduced to improved it. The research content has four parts. The first part briefly introduces the research topic of scheduling optimization models. The second part is to analyze the main methods used in this study. The third part analyzes the results. The fourth part is a summary for the study and prospects for future research.

II. Related Works

The scheduling model is a model established for scheduling problems. Ammari A C et al. proposed a scheduling strategy based on an improved firefly algorithm for delay constrained applications in distributed green data centers. Multiple heterogeneous applications were efficiently scheduled with less cost and energy. The proposed scheduling strategy model based on the improved firefly algorithm could meet the scheduling problem of distributed green data centers [3]. With the development of cloud and mobile applications, the integration demand for applications and services in business processes is also increasing. Many integrated platforms used heuristic algorithms to schedule tasks executed by computing resources. Therefore, Freire D L et al. proposed a queue priority algorithm. This algorithm was based on particle swarm optimization, which could handle massive amounts of data in integrated task scheduling. The algorithm could execute the integration process and schedule the data under high data volume [4]. Zhou J et al. found that crowd perception could solve the massive data collection faced by most data-driven applications. Therefore, a workflow framework was first proposed, which captured the unique execution logic of perception tasks. Then, a phased approach was proposed to decouple the original scheduling problem. From the experimental results, the proposed model had good performance in solving scheduling problems [5].

Mishra A et al. found that task scheduling was crucial for improving the performance of large-scale collaborative and distributed electronic science applications. Therefore, a meta-heuristic crow search algorithm was proposed to address the scheduling problem of multiple tasks across heterogeneous virtual machines. This method could demonstrate better model performance compared with traditional models [6]. The computing demand in various application fields is increasing day by day. To meet this requirement, on-site programmable gate arrays have been widely used. Therefore, Tianyang L et al. summarized the current research status of hardware task dynamic scheduling based on the three basic elements of

*Corresponding Author.

existing on-site programmable gate array processing: time, resources, and power consumption. The optimization effects of various scheduling methods were analyzed and evaluated from multiple dimensions. The research results indicated that the research could make a certain contribution to scheduling problems based on field programmable gate arrays [7]. Ye W et al. proposed a new unmanned aerial vehicle assisted edge computing system. The system dispatched edge nodes assisted by drones to provide communication and computational assistance for completing tasks generated by ground clients. Firstly, a trajectory design and task allocation problem were proposed, aiming to optimize the appropriate trajectory of each drone and schedule tasks for each ground client. A maximum drone trajectory and task allocation algorithm was proposed, which solved the task allocation problem by jointly optimizing the trajectory of the drone and the task scheduling of the ground client. The proposed method demonstrated good scheduling performance [8]. Wang et al. found that two-stage mixed flow workshop scheduling with batch machines and jobs arriving over time was complex and challenging. For online scheduling problems, traditional heuristic rules can quickly respond to dynamically arriving jobs, but their performance is poor and unstable. Therefore, a scheduling model based on the DQN algorithm was proposed. It transformed the online scheduling problem into a collaborative Markov decision process by defining the state space, action space, and reward function of different agents. The experimental results showed that the model could effectively combine online batch formation and scheduling, minimizing the total delay time [9]. Sun C et al. found that task scheduling and load balancing in heterogeneous computing environments received increasing attention in recent years. Therefore, a new task scheduling and load balancing method based on optimized deep reinforcement learning is proposed. This method first formulates the task scheduling problem into a Markov decision process. Then a dual deep Q-learning network was used to search for the optimal task allocation solution. The research results indicated that the proposed method model had shorter task response time and better load balancing effect [10].

In summary, many scholars have conducted research on task scheduling and achieved some results. In this study, a scheduling model based on Markov decision process is proposed, which applies DQN algorithm to DAG task scheduling. Then, to address the shortcomings of DQN algorithm, the PG algorithm is introduced to improve the model.

III. Big Data Task Scheduling Optimization Model based on Deep Q-network

The first section of this chapter provides an explanation for DAG task scheduling. The scheduling problem is optimized into a Markov decision model. A task scheduling algorithm based on DQN is proposed. In the second section, a PG algorithm is proposed to address the shortcomings of task scheduling algorithms based on DQN. Combined with the DQN algorithm, a scheduling model based on PG-DQN algorithm is proposed.

A. Directed Acyclic Graph Task Scheduling in Heterogeneous Environments

Cloud computing task scheduling refers to the rational allocation of tasks on cloud computing platforms to different computing resources, improving computing efficiency and resource utilization. Cloud computing servers usually have three parts, namely scheduling servers, work nodes, and data storage services. Performing computational tasks often requires the output of other tasks as input, which can be abstracted as a DAG representation. A DAG is a graph structure composed of nodes and directed edges. Each edge has a direction and there is no loop [11]. Starting from any node in the graph and following the direction of the directed edge, it will not return to that node. Its structure is shown in Fig. 1.

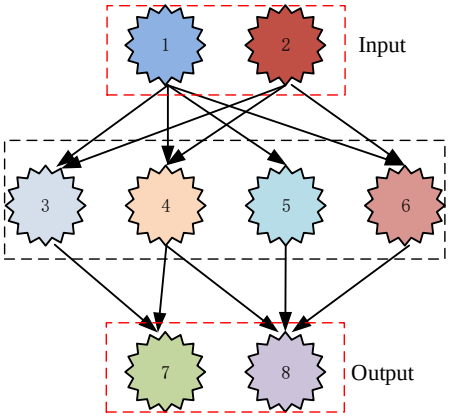


Fig. 1. DAG model diagram.

In Fig. 1, the node without a predecessor node task is the entry node, as shown in nodes 1 and 2 in the figure. There are no other nodes that rely on this node, such as node 7 and node 8. The computational cost of the task on each processor is shown in Eq. (1).

$$ECT_{V\times P}=\begin{bmatrix} ECT_{11} & \cdots & ECT_{1P} \\ \vdots & \ddots & \vdots \\ ECT_{V1} & \cdots & ECT_{VP} \end{bmatrix} \tag{1}$$

In Eq. (1), ECT represents the expected completion time. P refers to the computing node. V refers to the current task. The execution time is represented by the ECT matrix of $V\times P$, which is the time required to allocate each task to different nodes at the current moment [12]. In scheduling problems, the most common definitions are the earliest start time, earliest completion time, and maximum completion time. The earliest start time refers to the time when the task cannot start executing earlier than that, as shown in Eq. (2).

$$EST(v_i,p_j)=\max\left\{avail[j],\max_{v_k\in pred(v_i)}\{FT(v_k)+c_{k,i}\}\right\} \tag{2}$$

In Eq. (2), v_k represents the task. $FT(v_k)$ refers to the end time of the task execution. p_j represents a node. $avail[j]$ represents the earliest time used to calculate the

node. $c_{k,i}$ represents the communication overhead between two tasks. When the direct precursor of a node is on the same processor as the node, the communication overhead between the two nodes can be considered as zero [13]. The earliest completion time indicates that the task cannot be completed earlier than that time, as shown in Eq. (3).

$$EFT(v_i, p_j) = EST(v_i, p_j) + \omega_{i,j} \tag{3}$$

In Eq. (3), $EST(v_i, p_j)$ refers to the earliest start time of the task. $\omega_{i,j}$ represents the time required for the task to perform calculations on the computing node. The earliest completion time is equivalent to the sum of the earliest start time and task execution time of the task [14]. The maximum completion time represents the time required to complete the last task in DAG, as displayed in Eq. (4).

$$Makespan = \max\{FT(v_{exit})\} \tag{4}$$

In Eq. (4), *Makespan* represents the maximum completion time. v_{erit} represents the export task. The scheduling problem can be scheduled based on the execution time of the task. This process can be considered as a Markov decision process, which is a mathematical model used to describe stochastic decision problems. This method is based on an extension of Markov chain and decision theory. It is used to model sequential decision problems that include randomness and decision selection [15]. Its structure is shown in Fig. 2.

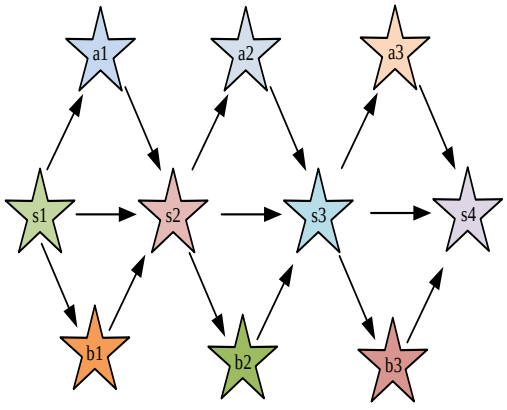


Fig. 2. Markov decision process.

From Fig. 2, the future state only depends on the current state and the currently selected action, rather than the past state and action. This nature makes the Markov decision process computable. Dynamic programming and other methods can be used to solve the optimal strategy [16-17]. The probability of a system transitioning from one state to another is defined as the state transition matrix, as shown in Eq. (5).

$$P = \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & & \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix} \tag{5}$$

In Eq. (5), P represents the state transition matrix. The goal of Markov decision process is to find a strategy that maximizes long-term cumulative rewards. The quality of a strategy is measured by defining a value function. It represents the long-term cumulative reward that can be obtained by adopting a certain strategy in a certain state, as shown in Eq. (6).

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{6}$$

In Eq. (6), R_t represents the reward obtained from the environment after taking the action. t represents time. γ represents the attenuation factor, which reflects the future returns on the current value of the intelligent agent [18]. If the return is far from the current moment, the attenuation will be greater. To measure the value of a state, the expected cumulative reward is used as the state value, as displayed in Eq. (7).

$$v_\pi = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \right] \tag{7}$$

In Eq. (7), π represents the probability distribution of taking action. s represents the state. γ represents the attenuation factor. S represents the finite set state of the system. Based on the Markov decision process, a model is established for task scheduling problems in heterogeneous environments. The specific scheduling process is shown in Fig. 3.

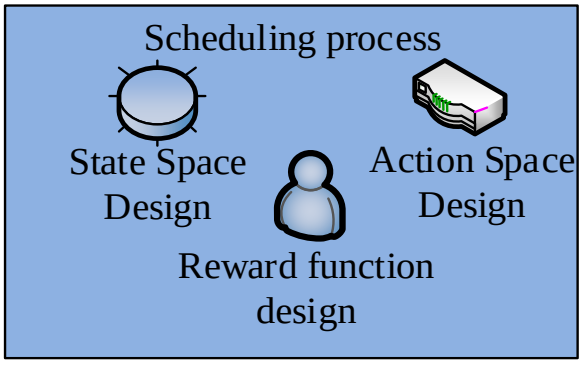


Fig. 3. Scheduling process.

The state space in a static environment is shown in Eq. (8).

$$S_t = [n, EST_1, \cdots, EST_M, T_{i,1}, \cdots, T_{i,M}] \tag{8}$$

In Eq. (8), t represents time. S_t represents the system state obtained by the scheduling model at t. M refers to the computing node. n refers to the current tasks. EST_j refers to the start time of the task in the processor. T represents the execution time. In task scheduling, the corresponding action space is shown in Eq. (9).

$$A_t = \{p_i | p_1, \cdots, p_M\} \tag{9}$$

In Eq. (9), A_t represents the action space. The model scheduling action at a certain moment is to schedule the current task to a computing node. The design of the reward function has significant impacts on the scheduling strategy. The designed reward function is shown in Eq. (10).

$$R_t = \max\{EST(v_i, p_j) \mid j = 1 \ldots q\} - \max\{EST(v_{i+1}, p_j) \mid j = 1 \ldots q\} \quad (10)$$

In Eq. (10), EST represents the start time of the task in the processor. In the Markov decision process, traditional methods lead to an extremely large state space, making modeling extremely slow and even impossible. Therefore, the reinforcement learning method is used to build models, which not only avoids the large state spaces, but also solves the continuous state spaces. DQN is used to build model. The core idea of DQN is to map the states and actions in the Markov decision process to a value function, namely the Q-value function. It is used to estimate the long-term return that can be obtained by taking an action in the current state. By learning this value function, DQN can select the optimal action in different states to maximize cumulative returns. DQN uses deep neural networks to approximate Q-value functions. The input of a neural network is a state. The output is an estimate of the Q-value for each action. After continuously adjusting the weights of the neural network, the estimation of Q value is closer to the true Q value. DQN uses an experience replay mechanism to train neural networks, which balances the sample correlation by saving and reusing previous experiences, improving training efficiency and stability.

The iterative process of the task scheduling algorithm based on DQN is as follows. Firstly, the set of tasks waiting for scheduling is inputted. The priority queue of the tasks is initialized. For tasks that meet the conditions, they will be arranged at the end of the queue. Secondly, the network parameters are initialized and weights are randomly generated. When there are tasks in the priority queue, the first task in the queue is selected as a pending scheduling task. Then the state is obtained. The action with the highest Q value is calculated. The task is scheduled to the corresponding computing node based on the action. Then, the return is calculated, and the DQN is backpropagated. Finally, the system status is updated.

B. Task Scheduling Optimization model Based on Deep Q-Network

The DQN algorithm has wide applicability and strong expressive ability. Combining deep learning with reinforcement learning, the DQN algorithm can be applied to various sequential decision problems, including game agent control, autonomous driving, robot path planning, etc. Therefore, it has high universality and flexibility. The DQN algorithm uses deep neural networks to approximate value functions, which can handle high-dimensional and complex state spaces [19]. Therefore, the DQN algorithm can provide better expressive power and performance when dealing with large-scale problems. The DQN algorithm combines deep learning and reinforcement learning, utilizing deep neural networks to learn the approximation of value functions. It can effectively deal with continuous state and action space problems, achieving good performance. The DQN algorithm uses experience replay and fixed target network methods to improve the sample utilization efficiency and training stability, avoiding the sample correlation and instability in traditional reinforcement learning methods.

However, the DQN algorithm model has some problems, such as the inability to represent random policies and difficulty in convergence. Therefore, the PG algorithm is adopted to improve the DQN algorithm. The PG algorithm is a method used to solve reinforcement learning problems by optimizing policy parameters to maximize cumulative rewards. Unlike traditional value function methods, the PG algorithm directly optimizes the policy function by using policy parameters as learnable parameters. The most commonly used method in PG algorithms is to use gradient ascent to update policy parameters. The core idea of the algorithm is to estimate the expected cumulative reward by sampling multiple trajectories. The gradient ascent method is used to update policy parameters to maximize expected rewards. To maximize cumulative rewards, a gradient ascent is applied to the cumulative rewards. The strategy function is shown in Eq. (11).

$$\pi(a \mid s, \theta) = P_r\{A_t = a \mid S_t = s, \theta_t = \theta\} \quad (11)$$

In Eq. (11), s represents the environmental state at time t. θ represents the model parameters. P_r represents the probability that a task is assigned to a computing node for execution. The task scheduling and model parameter optimization process is shown in Fig. 4.

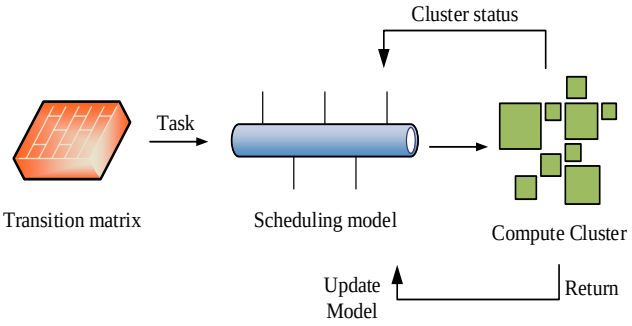


Fig. 4. Task scheduling and model parameter optimization process.

From Fig. 4, the scheduling model obtains a computing cluster through computing nodes. Then the computing cluster influences and updates the scheduling model through cluster status and returns [20]. For scheduling tasks, each task executes the above process to form a scheduling trajectory. The probability of generating scheduling trajectories is shown in Eq. (12).

$$P(\tau \mid \pi) = \rho_1(s_1) \prod_{t=0}^{T} P(s_{t+1} \mid s_t, a_t) \pi(a_t \mid s_t) \quad (12)$$

In Eq. (12), τ represents the scheduling trajectory. π represents the scheduling strategy. t represents the time step. The expected cumulative reward is shown in Eq. (13).

$$J(\pi_\theta) = \underset{\tau \sim \pi_\theta}{E}[R(\tau)] \quad (13)$$

In Eq. (13), $R(\tau)$ represents the cumulative reward. The training model needs to maximize the cumulative return. Therefore, the parameters are updated through the gradient ascent method. The iterative process based on PG is displayed in Fig. 5.

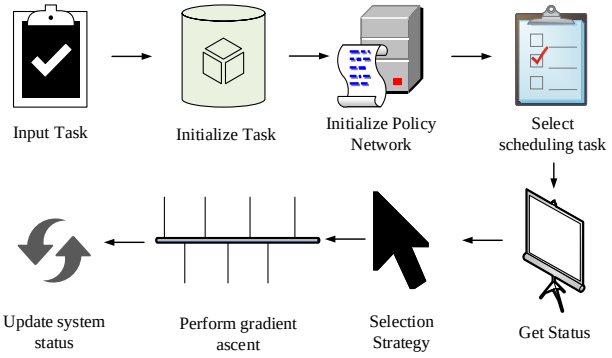


Fig. 5. The process of task scheduling algorithm based on PG.

In Fig. 5, the first step is to input a set of tasks waiting for scheduling, and initialize the priority queue of the tasks. For tasks that meet the conditions, they will be arranged at the end of the queue. Next, the policy network is initialized. If there are tasks in the priority queue, the first task in the queue is selected as a pending scheduling task. Then the state is obtained and the probability values for executing various scheduling actions are output. Based on the probability value, the current strategy action is selected. According to the policy action, the task is scheduled to the corresponding computing node. Then the reward of the scheduling action is calculated. Finally, the policy network is backpropagated and the system state is updated through the gradient ascent. The PG method is combined with the DQN algorithm to schedule tasks. The PG algorithm is responsible for outputting behavior, while the DQN algorithm evaluates behavior based on returns. Moreover, both the PG algorithm and the DQN algorithm simulate and update functions through neural networks. The combination of the two methods will result in better performance, as shown in Fig. 6.

In Fig. 6, all threads share a neural network, which includes the PG network and the DQN network. Each thread contains a neural network that is the same as the public network. Each network can be updated separately.

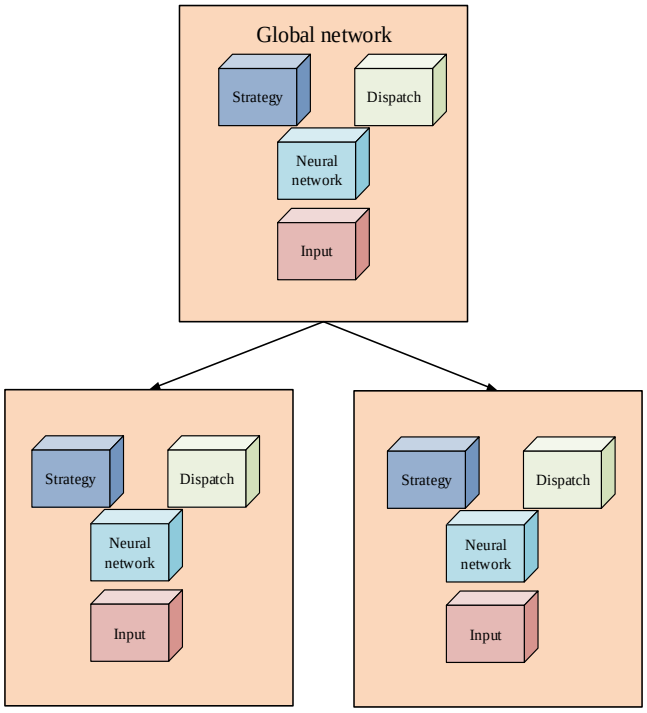


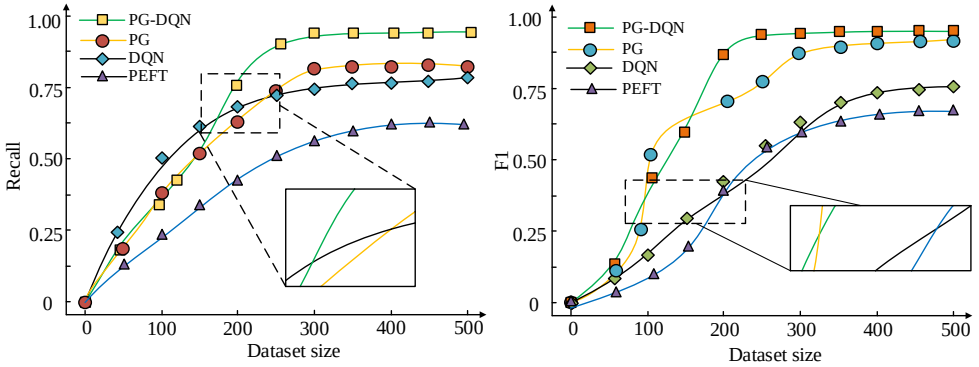
Fig. 6. PG-DQN network structure diagram.

IV. Performance Analysis of Big Data Task Scheduling Optimization Model Based on Deep Q-Network

The first section of this chapter introduces the Predict Earliest Finish Time (PEFT) algorithm. It is compared with the proposed algorithm. Chapter 2 compares the cumulative rewards and the maximum completion time for dynamic scheduling models.

A. Performance Analysis of Static Task Scheduling Optimization Model based on Improved Deep Q-network

The operating system used in this experiment is Windows 10. The CUP is the Intel Core i7-4710MQ processor, with a main frequency of 2.5GHz and 8.00GB of memory. The estimated earliest completion time algorithm and PG algorithm are compared with the algorithm used in this study [21-22]. The result is shown in Fig. 7.



(a)Recall rates under four algorithmic models

(b)F1 values under four algorithmic models

Fig. 7. Recall rates and F1 values of four methods.

Fig. 7(a) shows the recall rates of four algorithms on different datasets. Fig. 7(b) displays the F1 values of four algorithms on different datasets. In Fig. 7(a), the recall rates of the four models all increased with the increase of the training set. The proposed PG-DQN algorithm performed well among the four methods. When the dataset size was around 500, the recall rates of PG-DQN, PG, DQN, and PEFT were 0.96, 0.83, 0.79, and 0.63. In Fig. 7(b), the F1 values continued to increase with the increase of the training set. When the dataset size was 500, the F1 values of the four algorithms were 0.97, 0.90, 0.76, and 0.65, respectively. The proposed PG-DQN exhibits good performance in terms of recall and F1 value among the four models. Moreover, the PG-DQN has good performance on smaller datasets. The scheduling length ratio in the scheduling task is used as an indicator for comparison. Fig. 8 displays the results.

In Fig. 8, compared with the traditional PEFT, DQN and PG had a smaller scheduling length ratio. The proposed PG-DQN had a smaller scheduling length ratio than the other three algorithms when scheduling 200-1000 tasks. The PG-DQN can explore more reward actions. The proposed PG-DQN algorithm performs well. It can still maintain good performance in multiple tasks. The training time and scheduling time are compared. Fig. 9 displays the results.

Fig. 9 (a) shows the training time on different datasets and Fig. 9(b) presents the processing time of the algorithm under different task quantities. From Fig. 9(a), the FEPT exhibited longer training time under different dataset sizes. The training time of the PG-DQN was slightly longer than that of the PG and slightly shorter than that of the DQN. This is because the proposed algorithm model is a hybrid model of two methods, resulting in a more complex structure and more calculated parameters. In Fig. 9(b), the proposed PG-DQN only had slightly higher processing time than the PG in various quantities of task scheduling. Although the proposed PG-DQN does not take the least time, considering the scheduling performance, the overall performance of the PG-DQN is still better.

B. *Performance Analysis of Dynamic Task Scheduling Optimization model based on Improved Deep Q-network*

The experiment randomly generates 100 DAG tasks, each containing 10 sub-tasks, which are used as the dataset. The result is shown in Fig. 10.

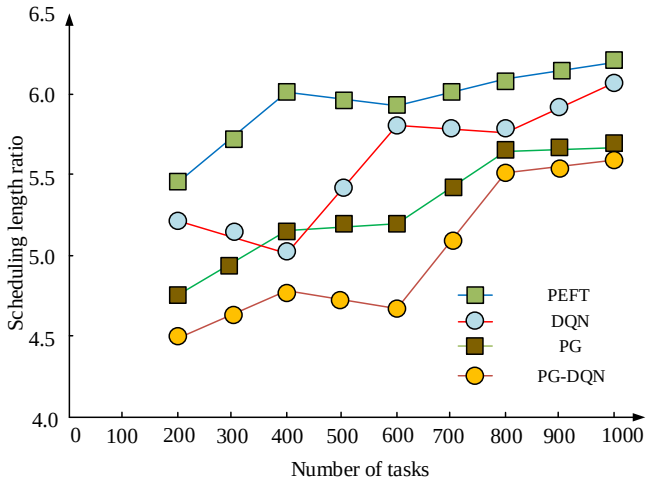
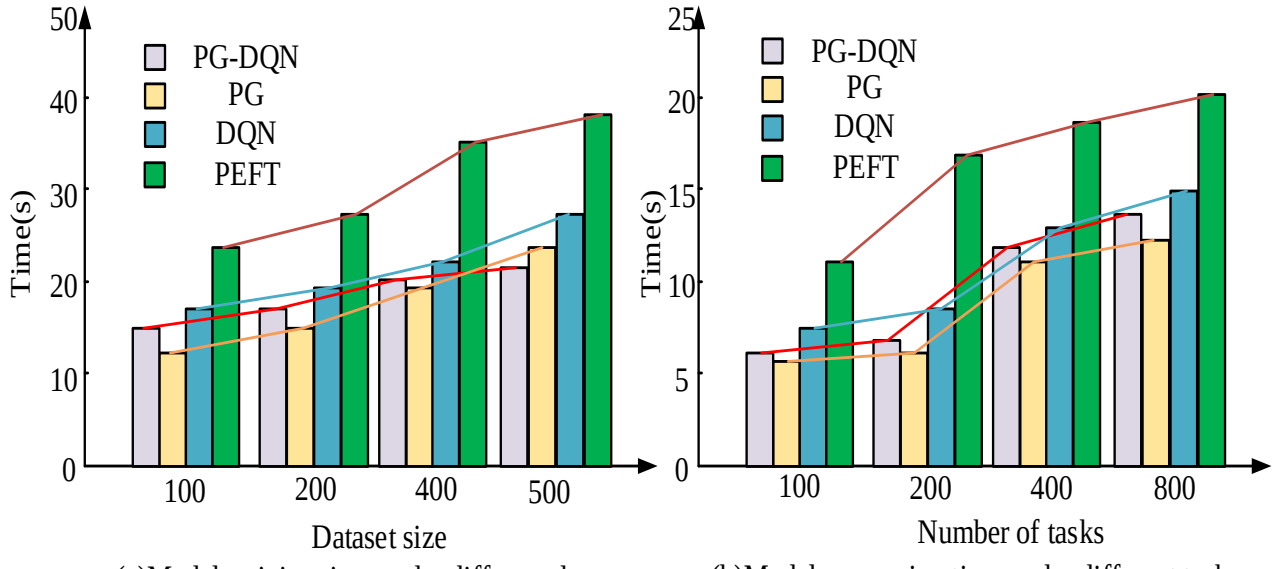
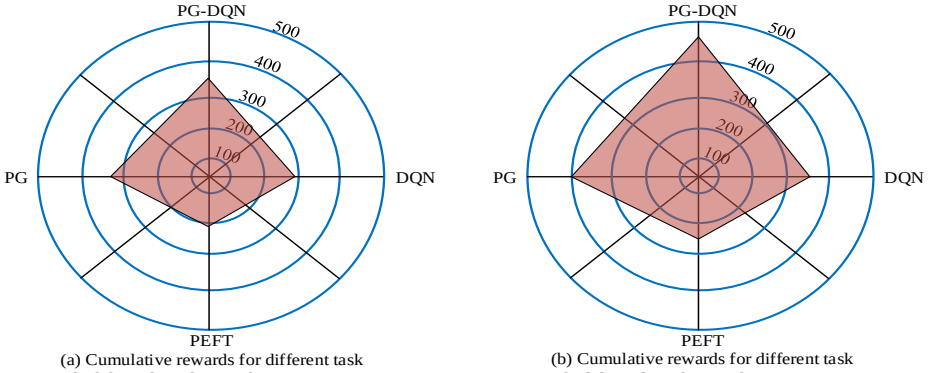


Fig. 8. The relationship between the scheduling length ratio index of different algorithms and the number of tasks.



(a)Model training time under different datasets (b)Model processing time under different task quantities

Fig. 9. Calculation time required for scheduling different algorithms.



(a) Cumulative rewards for different task scheduling algorithms with 100 iterations

(b) Cumulative rewards for different task scheduling algorithms with 500 iterations

Fig. 10. Cumulative rewards of task scheduling algorithm under different iterations.

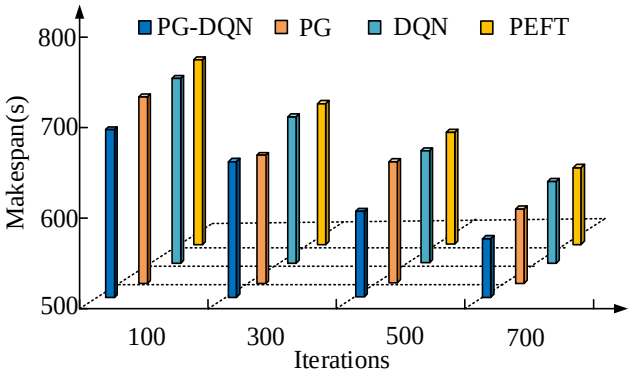
Fig. 10(a) presents the cumulative rewards of different task scheduling algorithms at 100 iterations. Fig. 10(b) displays the cumulative rewards of different task scheduling algorithms at 500 iterations. In Fig. 10(a), at 100 iterations, the cumulative reward for the PG-DQN, PG, DQN, and PEFT were 350, 320, 290, and 210, respectively. In Figure 10 (b), at 500 iterations, the cumulative reward for the PG-DQN, PG, DQN, and PEFT were 460, 400, 350, and 260, respectively. The cumulative return of the model is lower when the number of iterations is small. After reaching 500 iterations, the cumulative return of the model is good. The proposed PG-DQN has the highest cumulative return among the four models, indicating that the PG-DQN has good performance. The maximum completion time is compared, as displayed in Fig. 11.

Fig. 11(a) displays the maximum completion time at different iterations. Fig. 11(b) displays the maximum completion time at different task quantities. According to Fig. 11 (a), when the number of iterations was 100, 300, 500, and 700, the maximum completion time of PG-DQN was 723s, 654s, 591s, and 576s, respectively, which were lower than the other three algorithm models. In Fig. 11(b), when the scheduling tasks were 100, 200, 300, and 400, the maximum completion time of PG-DQN was 342s, 387s, 410 s, and 442s, respectively. The proposed algorithm model has good model performance at different iterations and task quantities. 50 users are randomly selected and divided into an average of five groups to rate the model, as shown in Table I.

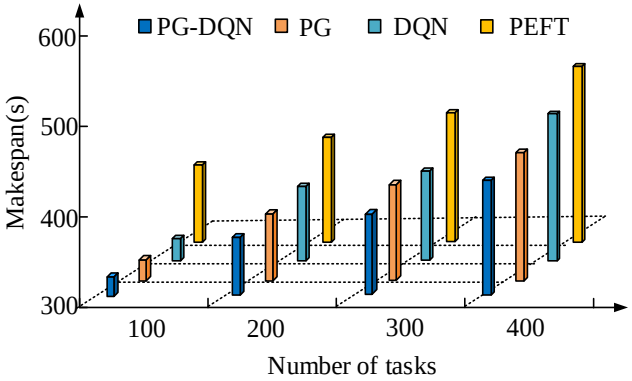
According to Table I, the scores of the five groups on the PG-DQN were 84.4, 97.2, 92.5, 94.7, and 90.1, respectively. The scores for the PG were 80.2, 94.5, 90.4, 90.6, and 86.5, respectively. The scores for the DQN were 78.6, 80.7, 85.4, 87.6, and 82.4, respectively. The scores for the PEFT were 76.8, 78.3, 82.4, 86.1, and 78.1. The PG-DQN has better scores among the four models, which has received widespread praise.

TABLE I. USER EVALUATION FORM

/	Group 1	Group 2	Group 3	Group 4	Group 5
PG-DQN	84.4	97.2	92.5	94.7	90.1
PG	80.2	94.5	90.4	90.6	86.5
DQN	78.6	80.7	85.4	87.6	82.4
PEFT	76.8	78.3	82.4	86.1	78.1



(a)Makespan of four algorithms under different iterations



(b)Makespan of four algorithms under different task quantities

Fig. 11. Comparison of maximum completion time for tasks.

V. CONCLUSION

The task scheduling algorithm has significant value for the platform operation efficiency. A scheduling model based on Markov decision process is proposed, which applies DQN algorithm to DAG task scheduling. Then, to address the shortcomings of DQN algorithm, the PG algorithm is introduced to improve the model. According to the results, the recall rates of the four models increased with the increase of the training set. When the dataset size was around 500, the recall rates of the PG-DQN, PG, DQN, and PEFT were 0.96, 0.83, 0.79, and 0.63. The F1 values of the four algorithms were 0.97, 0.90, 0.76, and 0.65. Under different dataset sizes,

the training time of the PG-DQN was slightly longer than that of the PG and slightly shorter than that of the DQN. In various task scheduling quantities, the proposed PG-DQN only had slightly higher processing time than the PG. At 100 iterations, the cumulative reward for the PG-DQN, PG, DQN, and PEFT were 350, 320, 290, and 210, respectively. At 500 iterations, the cumulative reward for the PG-DQN, PG, DQN, and PEFT were 460, 400, 350, and 260, respectively. The proposed method has good scheduling performance. However, there are also shortcomings in the research. The study only considers the execution time prediction for single threaded tasks. Future research will be conducted on multi-threaded tasks. For the task scheduling model, in the future, parameters will be adjusted based on the existing foundation. The model structure will continue to be optimized to achieve better scheduling results. The model architecture will also be considered for multi-objective optimization, such as fairness between tasks and priority scheduling of tasks.

VI. Discussion

The scheduling model is a model established for scheduling problems. This study proposes a scheduling model based on Markov decision process, which applies the DQN algorithm to DAG task scheduling. Then, to address the shortcomings of the DQN algorithm, a PG algorithm is introduced to combine it and improve the model. The experimental results showed that the DQN algorithm had a smaller scheduling length ratio compared with the traditional PEFT algorithm and PG algorithm. The proposed PG-DQN algorithm had a smaller scheduling length ratio than the other three algorithm models when scheduling 200-1000 tasks, indicating that the PG-DQN algorithm model can explore more rewarding actions. Under different dataset sizes, the FEPT algorithm exhibits longer training times. The training time of PG-DQN algorithm model was slightly longer than that of PG algorithm and slightly shorter than that of DQN algorithm. This is because the proposed algorithm is a hybrid model of two methods, resulting in a more complex structure and more calculated parameters. When the number of iterations were 100, 300, 500, and 700, the maximum completion time of PG-DQN was 723s, 654s, 591s, and 576s, respectively, which were lower than the other three models. When the scheduling tasks were 100, 200, 300, and 400, the maximum completion time of PG-DQN was 342s, 387s, 410 s, and 442s, respectively. The research results indicated that the proposed method model has better performance and efficiency.

Fundings

The research is supported by National Social Science Foundation Project "Research and Application of Key Technologies for Intelligent Ideological and Political Classroom in Universities Based on Big Data" (No.19VSZ084); Chongqing Natural Science Foundation Project "Collaborative Mining and Online Classification Method Research for Multi-Source Partial Label Big Data in the Federated Learning Framework" (No. CSTB2022NSCQ-MSX1421).

Conflict of Interest

The authors have no relevant financial or non-financial interests to disclose.

References

[1] Dai X, Zhao L, Li Z, Du W, Zhong W, He R, Qian F. A data-driven approach for crude oil scheduling optimization under product yield uncertainty. Chemical Engineering Science, 2021, 246(32):124-133.

[2] Jiang J. Intelligent City Traffic Scheduling Optimization Based on Internet of Things Communication. Wireless Communications and Mobile Computing, 2021, 10(2):1-10.

[3] Zhang Z L, Zhang H J, Xie B, Zhang X. Energy scheduling optimization of the integrated energy system with ground source heat pumps. Journal of cleaner production, 2022, 365(10):1-19.

[4] Ammari A C, Labidi W, Mnif F, Yuan H, Zhou M, Sarrab M. Firefly algorithm and learning-based geographical task scheduling for operational cost minimization in distributed green data centers. Neurocomputing, 2022,490(14):146-162.

[5] Freire D L, Frantz R Z, Roos-Frantz F, Basto-Fernandes V. Queue-priority optimized algorithm: a novel task scheduling for runtime systems of application integration platforms. Journal of supercomputing, 2022, 78(1):1501-1531.

[6] Zhou J, Fan J, Wang J. Task scheduling for mobile edge computing enabled crowd sensing applications. International Journal of Sensor Networks, 2021, 35(2):323-329.

[7] Mishra A, Sahoo M N, Satpathy A. H3CSA: A makespan aware task scheduling technique for cloud environments. Transactions on Emerging Telecommunications Technologies, 2021, 32(10):381-397.

[8] Tianyang L, Fan Z, Wei G, Sun M, Chen L. A Survey: FPGA-Based Dynamic Scheduling of Hardware Tasks. Chinese Journal of Electronics, 2021, 30(6):991-1007.

[9] Wang M, Zhang J, Zhang P, Cui L, Zhang G. Independent double DQN-based multi-agent reinforcement learning approach for online two-stage hybrid flow shop scheduling with batch machines.Journal of Manufacturing Systems, 2022, 65(32):694-708.

[10] Sun C, Yang T, Lei Y. DDDQN-TS: A task scheduling and load balancing method based on optimized deep reinforcement learning in heterogeneous computing environment.International journal of intelligent systems, 2022, 37(11):9138-9172.

[11] Ye W, Luo J, Wu W, Shan F, Yang M. MUTAA: An online trajectory optimization and task scheduling for UAV-aided edge computing. Computer networks, 2022, 218(9):1-13.

[12] Gordon C A K, Pistikopoulos E N. Data-driven prescriptive maintenance toward fault-tolerant multiparametric control. AIChE Journal, 2022, 68(6):1745-1761.

[13] Deng Q, Santos B F. Lookahead approximate dynamic programming for stochastic aircraft maintenance check scheduling optimization. European Journal of Operational Research, 2022, 299(3):814-833.

[14] Gao Z, Sun D, Zhao R, Dong Y. Ship-unloading scheduling optimization for a steel plant. Information Sciences, 2021, 544(21):214-226.

[15] Du H, Zhang K, Xiang Q. Stargazer: Toward efficient data analytics scheduling via task completion time inference. Computers & Electrical Engineering, 2021, 92(8):1070-1092.

[16] Gil-Mena A J, Bouakkaz A, Salim H. Online Load-Scheduling Strategy and Sizing Optimization for a Stand-Alone Hybrid System. Journal of Energy Engineering, 2021, 147(1):431-442.

[17] Maiorino A, Mota-Babiloni A, Manuel D, Ciro A. Scheduling Optimization of a Cabinet Refrigerator Incorporating a Phase Change Material to Reduce Its Indirect Environmental Impact. Energies, 2021, 14(8):241-252.

[18] David M, Boland J, Cirocco L, Lauret P, Voyant C. Value of deterministic day-ahead forecasts of PV generation in PV + Storage operation for the Australian electricity market. Solar Energy, 2021, 224(8):672-684.

[19] Zhang J, Xing L. An improved genetic algorithm for the integrated satellite imaging and data transmission scheduling problem. Computers & operations research, 2022, 139(5):1392-1405.

[20] Mehdi G, Hooman H, Liu Y, Peyman S, Arif S. Data Mining Techniques for Web Mining: A Survey, Artificial Intelligence and Applications, 2022, 1(1):1-13

[21] Liang J, Li K, Liu C, Li K. Are task mappings with the highest frequency of servers so good? A case study on Heterogeneous Earliest Finish Time (HEFT) algorithm.Journal of systems architecture, 2021, 121(41):1355-1362.

[22] Weerakody P B, Wong K W, Wang G. Policy gradient empowered LSTM with dynamic skips for irregular time series data.Appl. Soft Comput. 2023, 142(21):110314-110321.