2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)

2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS) | 978-1-7281-7649-9/20/$31.00 ©2020 IEEE | DOI: 10.1109/HPCC-SmartCity-DSS50907.2020.00067

An improved reinforcement learning approach to solve flow job scheduling problems

1st Dapeng Hu
College of Computer Science and Technology
Qilu University of Technology
(shandong academy of sciences)
Jinan, China
hudapeng2017@163.com

2nd Xuesong Jiang*
College of Computer Science and Technology
Qilu University of Technology
(shandong academy of sciences)
Jinan, China
jxs@qlu.edu.cn

3rdJian Wang
Software and Bigdata Department
Shandong College of Information Technology
Weifang,China
wj7059830570@sina.com

***Abstract*—The fow shop scheduling problem is an important scheduling problems with a large number of practical applications. For the high performance computing applications, flow shop scheduling directly affects resource utilization and power dissipation.In this paper, we propose a novel algorithm TS_Qlearning algorithm that combines the tabu search(TS) method and the Qlearning algorithm to minimize the idle time. We also calculate the makespan value ,which is used to comprehensively evaluate the quality of the algorithm. The comparative analysis of the results proves the superiority of the TS_Qlearning algorithm.**

***Index Terms*—flowshop; Qlearning;scheduling; reinforcement learning**

I. Introduction

High performance computing refers to equipment with powerful computing capabilities to complete the specified computing workload in a short period of time.With the rapid growth of data, the demand for computer resources is also higher, which puts forward higher and more comprehensive requirements for data storage, calculation speed and response time.In the face of such a large amount of data workload, it is obviously limited to improve the hardware configuration to achieve computing performance [1]. Therefore, balancing the load of each node by flow shop scheduling, improving the computing speed, and improving the reasonable allocation of resources have become the focus of high performance computing research.

Flow shop scheduling problem(FSSP) is considered as an NP problem, as it is hard to find a feasible solution in polynomial time.FSSP is simply that there are m uncorrelated machines and n jobs [2].For every job must be made up of m operation groups running on different machines. All tasks pass through the machine in the same order. There is no priority restriction between different task operations. Operations don't allowed to be interrupted; ever machine is able to operate one at a time [5].The goal is to find a job sequence on the machine that minimizes idle time.

Recently, many researchers [6] have used the Qlearning algorithm which is more effective to solve FSSP problem. The Qlearning algorithm is a classic algorithm in reinforcement learning. Originally developed by Sutton and Barto [3], reinforcement learning is actually an area of machine learning, but it is different from our common machine learning (such as supervised learning).Reinforcement learning is tracking and error learning. Because of indirect guidance, agents must constantly communicate to the environment in order to get the best strategy in training.With the continuous learning of agents, the method will obtain more and more close to the optimal solution even in the case of changing environment .And it takes advantage of the high performance computing of artificial intelligence algorithms.However, a disadvantage of the Qlearning algorithm is that it does not know what action should be taken in an unknown state [4]. In other words, the Qlearning agent cannot evaluate the unknown state. This is likely to happen in the early stages of training.

The crucial contributions are as follows:1.An improved reinforcement learning algorithm of TS_Qlearning is proposed to solve flow shop problems.2.A "memory table" is established through the tabu table of the tabu search algorithm to store the early training experience of the algorithm to guide the early training of the algorithm, and to better select the initial solution in the early training of the algorithm.3.Changing the policy of the Qlearning algorithm,the superiority of the

Qlearning algorithm are retained and the algorithm training is guided in early state,thereby increasing the quality of the algorithm training.After gaining the trained results,we will also calculate makespan(C_{max}) value to comprehensively evaluate the quality of the algorithm.

II. LITERATURE REVIEW

The work of this article is based on previous work, including high performance computing and the FSSP.

Xiaokang Wang, et al. [7], [8] presented a general model of tensor calculation, which reduced execution time and optimized energy of tensor calculation under the premise of safety and reliability.In addition,they proposed the tensor method to enhance the feature recognition, identify the target from the angle of multiple viewpoints, in order to achieve the purpose of industrial intelligence.

With the development of current deep learning technology(DRL), Xiaofei Wang,et al. [9] proposed to combine the DRL technology and changeable frame with the mobile edge system to make the mobile edge system more intelligent than the traditional optimization methods.

Lifeng Nai,et al. [11]presented Graphpim, a complete stack solution for graphic computing and got better performance by using PIM function.Hu Zhu,ea al. [12]proposed a hypergraph adapting model with an entropy function to improve the efficiency of the subsystem.

Cai Z,et al. [13] established two models, one centered on the task owner and one centered on the mobile user to exploit job variety and improve performance of MCSs.Multiple heterogeneous MEC servers, and the model of ”minimum energy consumption problem in deadline-aware MEC system ” was established [14],and two approximate algorithms were proposed for single and multiple MD scenarios.

Lei Ren,et al. [15] proposed an evolutionary algorithm for generating coding groups. This indicated that operators of different servers was able to assign large tasks without the distribution of multi-device connection attributes of edge servers. If experimental results were included in the three cases, it provided an algorithm to solve large-scale tasks in a short time with the previous method.Using deep learning technology for recognition based on sensory data, the prediction accuracy of convolutional neural network for target location inference reached a high level [16], [17].

Fonseca-Reyna, Y. C., Martínez-Jiménez, Y.and Nowé, A [6] and Stefan et al. [18]used Qlearning to reduce the idle time or makespan. The performed analysis proved the superiority of the Qlearning.Henneberg, M.and Neufeld, J. S. [19] modified NPS-set heuristic to enhance the algorithms'performance significantly.

Z. Duan, et al. [20] designed a distributed auction framework. Then, based on this framework, proposed two kinds of distributed auction schemes: cost first auction and time first auction, to improve the efficiency of task allocation, the utilization of working time and the utility of mobile users.In order to allocate jobs to the edge server with the minimum completion time, Fang X, et al. [21] proposed the lower bound approximate offline algorithm in the edge computing environment, which also applied to the online scene. Compared with the optimal solution, both offline and online algorithms obtained good performance.

Liu et al. [22] proposed an optimized evolution algorithm L-HDE .In order to fit LDE to PFSSP, a novel maximum rank rule (LRV) based on random key was introduced to solve the continuous position in De ,which was converted into discrete work arrangement and the local minimum got rid of by combining local greedy local searchand.By covering the user's abnormal behavior to protect privacy, a new data uploading mechanism was proposed, which saved energy and protectd users' privacy [23].

III. BACKGROUND AND METHODS

A. SCHEDULING TASK

In this paper,the scheduling problem is considered as the flow shop scheduling. Groups with m machines and n tasks. Every job contains a series of m operations that must be run on a different machine. Each job is delivered from the machine in the same order.There is no priority restriction between different task operations. Operations cannot be interrupted; each machine is able to operate one at a time.For all machines, each part has the same process path and the order of work is arbitrary. The goal is to find an appropriate sequence of operations to minimize the sum of idle time.

In addition, in order to reflect the actual time required to complete all the jobs, we also calculated the value of makespan or C_{max} to reflect the rationality of the results. If the final result reduces the value of idle time and actually requires more time to complete this task sequence, this is obviously unreasonable.

Based on Stefan 's method [17],we calculate idle time. We introduce idle time using an example.There are 4 jobs that have to be performed on 3 unrelated machines.We assume that the obtained job sequence is $\{J_A, J_B, J_C, J_D\}$. $J_i m_j$ refers to the time that job J_i needs to spend on machine j .Figure 1 demonstrates a Gantt chart of a possible job sequence.

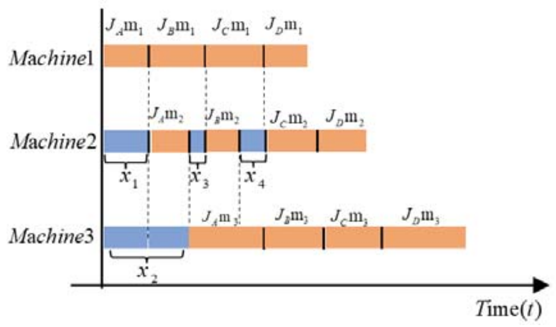

Fig. 1. Job sequence on three-machines

In the figure, $X = \sum_{i=1}^{n} x_i$ indicates the total idle time of the scheduling task with three-machines. The goal of optimization is to find a job-order, which minimizes X . In this paper, we define the idle time outside the Gantt chart as the "external idle time", such as x_1, x_2 .The others are defined as "internal idle time" such as x_3, x_4. In fact,the value of "external idle time" is directly determined by the initial solution, while the value of "internal idle time" is determined by the final solution.The details will be described in the following section D.

The general flow shop problem is represented in the literature as$n/m/C_{max}$ [10] for n jobs, each of which requires action on an m machine.In Python, we define the n-dimensional matrix p and C , whose row number are n and column number are m. We get the processing time $p(J_i, j)$ of job J_i on machine j through the dataset and a job queue $\{J_1, J_2, ..., J_n\}$, then we calculate the finished times $C(J_i, j)$ by Šeda [24] function.

Therefore, when the job permutation is $\{J_1, J_2, ..., J_n\}$, C_{max} is the time when the last operation of job J_n is achieved. Because the flow shop scheduling problem is NP problem,which consumes a lot of resources.We only bring the final job permutation trained by Qlearning algorithm or TS_Qlearning algorithm into the above formula to obtain C_{max}.

B. REINFORCEMENT LEARNING

Reinforcement learning [25] is an critical machine learning method.Reinforcement learning is tracking and error learning and a obvious difference is that reinforcement learning has no direct instructional information, so agents must constantly experiment to communicate to the environment that gets the best strategy.

In a series of cases, it is a continuous multi-stage decision problem to achieve the goal through multi-stage appropriate decision. The purpose of reinforcement learning is to solve the problem of how the autonomous learning agent perceives the environment and chooses the best action to achieve the goal through learning.The subject of reinforcement learning agents is the mapping of learning from environment to action.In reinforcement learning, the enhancement signals provided by the environment are not to assess the quality of the agent's operations, but to indicate how the agent generates the correct operations.Reinforcement learning mainly includes four elements: agent, environmental state, action and reward.The goal of reinforcement learning is to maximize rewards. [26].

For every step of interacting with the environment, the agent senses the current state in the environment and selects the role to change the status. After the conversion, the agent receives a reward signal. The agent task is to select the strategy that maximizes the long-term cumulative reward and select the actions for each state. The RL method will explore the environment over time, then propose the required strategy. The change of state is only related to the previous state and the current action, while not related to the previous state and action (that is, to satisfy Markov's property). Then the entire process can be described by Markov Decision Processes (MDP) .The finite Markov decision process consists of four groups $M = (S, A, P, R)$. As mentioned above, $S$is the state set space, A is a collection of actions, P is a matrix to describe state transition probability, and R is the expected earnings value. In MDP, when given a state $s \in S$ and action $a \in A$, the reward value $R(s, a, s')$ is calculated by moving to the next state s' with the probability $P(s'|s, a)$.

Reinforcement learning is able to be combined with supervised learning. For example, agents can be trained through supervised learning. In supervised learning, the subject should be shown a mapping of states and actions and indicated that the actions are correct or incorrect.The goal of supervised learning is to generalize general strategies from training examples. Therefore, supervised learning needs to provide a training sets of properly marked examples. In contrast, RL does not need to determine the correct and wrong knowledge of the training sets, and RL can be applied in situations where rewards are scarce.After the learning phase, each agent reacts intentionally to its environment. For the reward, what will happen in the future depends on the current situation and behavior of the environment. Most reinforcement learning researchers have been concerned about learning in this environment, and put forward many important reinforcement learning algorithm such as the Qlearning algorithm [27].

C. QLEARNING

Qlearning algorithm is an effective reinforcement learning algorithm. The target is to achieve the goal state and get the maximal reward. In case of that hits the goal state, the best reward doesn't change. The Qlearning algorithm agent does not know the entire environment, but is able to choose which action in the current state.Reinforcement learning usually requires the construction of a real-time reward matrix R [28] that represents the action reward value from one state to the next state s'. Qlearning is a very powerful algorithm, and its main disadvantage is the lack of generality. This means that an agent trained by the Qlearning algorithm does not know what to do in an invisible state

Qlearning perform an action value mapping function for current policy π as $Q^{\pi}(s, a) = E[R|s' = s, a' = a]$.

The update function for the state action is as follows:

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[r + \gamma max_a' Q(s', a')] \quad (1)$$

In EQ.1,α is defined as the learning rate and γ is the reward or punishment for performing action a.

In the state,the learning rate α determines 'the level' by which the obtained value is updated. γ is discount factor.

When a Qlearning method is applied to a specific problem, the attributes that can be used as state should be identified. In addition, state changes and reward should be identified. Then, the Qlearning algorithm explores and traverses the state space, trying to find a path to obtain higher reward. The advantage of the Qlearning algorithm lies in its exploration ability, that is, by bypassing the local maximum, it can traverse the state that have no good returns but may obtain higher reward in the long run.

In the flow shop scheduling training, we take machines time and processing cost as input parameters and job sequence as variable parameters,.The goal is to obtain an appropriate sequence of operations to minimize the value of idle time.

In order to adapt to RL method, state can be defined as job sequence reasonably, or job priority relationship more accurately. A state change (or action) is defined as a change in a job precedence relationship. At the beginning, no preferences are given, therefore states are selected randomly. With the development of learning proceeds, preferences are updated, which, influences action selection policy converging to finding optimal job sequence.

D. TS_QLEARNING

We assume that n jobs need to be executed on m unrelated machines, and through the training of the algorithm, we obtain the final job sequence. Similarly, we assume that J_k is the first job in the final job sequence. Then $J_k m_i$ represents the time that job J_k needs to spend on machine m_i .In fact, J_k is the initial solution obtained by the algorithm. Through this initial solution , we can obtain the value of "external idle time".The calculation of "internal idle time" requires a complete solution.

We define idle time as T , external idle time as T_e , and internal idle time as T_i .Obviously, the total idle time is equal to the sum of T_e and T_i . T_e can be obtained according to the following formula:

$$T_e = (m-1)J_k m_1 + (m-2)J_k m_2 + ... + J_k m_{m-1} \quad (2)$$

According to this formula, we can draw the conclusion that external idle time is only related to the initial solution, and the more the number of machines, the greater the impact on the total number.

A disadvantage of the Qlearning algorithm is that it does not know what action should be taken in the invisible state. In other words, the Qlearning agent has no ability to assess the unknown state. This is very likely to happen in the early stages of training.

To solve this problem, we present TS_Qlearning algorithm that combines the TS algorithm and the Qlearning algorithm.In this algorithm, we record some better initial solutions through tabu list. It is noted that we do not acquire the best initial solution through TS algorithm. On the contrary, we regard it as a "memory table" and exclude some bad solutions, which will cause very large external idle time.

In order not to affect the superiority of the Qlearning algorithm, the "memory table" policy of TS_Qlearning is only used in the early stage of training. Through this policy, the algorithm enable avoid the problem of poor results in the early stage of Qlearning training.It is noted that when using the "memory table" policy training, the Q table should also be updated with EQ.1, so that these excellent initial solutions in the early training will be recorded by the Q table, which will affect the subsequent training.

In addition, when the dataset and the length of tabu table are determined, the memory table is also determined.The memory table can be obtained by Tabu search before algorithm training, and it does not affect the convergence speed of the algorithm.The following describes the specific steps of TS_Qlearning algorithm.

Since $J_k m_1$ has the largest coefficient in external idle time function, $J_k m_1$ has the greatest influence on external idle time. Therefore, we define the candidate solution as $\{J_1 m_1, J_2 m_1, ..., J_k m_1, ..., J_n m_1\}$ .We put the smaller solution Job in the candidate solution into the tabu list until the number of iterations reached. We can adjust the length of the tabu table to control the scope of the initial solution.

Similarly, in the flow shop scheduling training, the machine time and processing cost are used as input parameters, and the job sequence is used as a variable parameter. The goal is to find a suitable job sequence to minimize the idle time.The TS_Qlearning algorithm is summarize as Algorithm 1.

Algorithm 1 TS_Qlearning algorithm for FSSP

```
1:  INITIALIZE:
2:  $Q(s,a) = \{\}$
3:  $best = \{\}$
4:  for $each\ episode\ step\ do$ do
5:    while $not\_finished(all\ jobs)$ do
6:      INITIALIZE:
7:      $s = \{\}$
8:      $job\_seq = \{\}$
9:      if $job\_seq\ is\ null\ \ \&\ episode < maxepisode/3$ then
10:       $choose\ a\ randomly\ from\ tabu\_list$
11:       $Take\ action\ a\ ,\ observe\ state\ s'\ and\ r$
12:       $Q(s,a) \leftarrow (1-\alpha)Q(s,a)+\alpha[r+\gamma max'_a Q(s',a')]$
13:       $s \leftarrow s'$
14:       $job\_seq \leftarrow job\_seq + s'$
15:     else
16:       $choose\ a\ from\ s\ using\ policy\ derived\ from Q(\epsilon-greedy)$
17:       $Take\ action\ a\ ,\ observe\ state\ s'\ and\ r$
18:       $Q(s,a) \leftarrow (1-\alpha)Q(s,a)+\alpha[r+\gamma max'_a Q(s',a')]$
19:       $s \leftarrow s'$
20:       $job\_seq \leftarrow job\_seq + s'$
21:     end if
22:   end while
23:   if $idletime(s) < idletime(Best)$ then
24:     $best \leftarrow s$
25:   end if
26: end for
```

To adapt to RL method, state can be defined as job sequence reasonably, or job priority relationship more accurately. A state change (or action) is defined as a change in a job precedence relationship.The initial solution of the

TS_Qlearning,which is different from Qlearning, is obtained randomly from the memory table(tabu list). The solutions in this "memory table" is able to be placed in the Q table early in the training, which guides to the direction of the Qlearning in the early training.

It is important to note that TS_Qlearning selects an initial solution, which is also equivalent to performing an action. Similarly, after executing the action, we will also obtain reward, the next state and the updated the Q table. When solving the scheduling problem [29], different feedback signals can be used.We use idle time as a reward signal, which means the lower the idle time, the better the action.

In this method, we obtain the required tabu list(memory table) before training. With the training going on, the preference will be updated constantly, which will affect the behavior selection strategy to converge to the quasi optimal job sequence. When the training is finished, we obtain the final job sequence and the total idle time, and then obtain C_{max}according to the C_{max} calculation formula.

In fact, we interpret the flow shop scheduling problem as a continuous decision making process, which is equivalent to attaching an adaptive agent to each resource, making its scheduling decision independent, and improving its scheduling behavior through trial and error of reinforcement learning algorithm.

IV. EXPERIMENT

Qlearning algorithm has been proved to be an efficient algorithm to solve scheduling problems. In those instances, the comparison of Qlearning method to solve scheduling problems with other traditional methods (such as genetic algorithm [30],ant colony optimization algorithm [31],and tabu search algorithm [32]) shows that the Qlearning method is able to obtain the best value in most instances [33].

In this part,we used the basic scheduling benchmark instances available in the OR-Library [34]. The OR library is an operational research problem that collects test data sets.There are n jobs that need to be executed on m independent machines.At this point, each task consists of m non-preemptive operations.Each operation of a job is able to stand idle until it is processed by another machine at a specified time. Scheduling instances provided three types of instances, and we used data from two instances of flow shop sequencing Job shop scheduling, which were proposed by E. Taillard [35]. These datasets introduced the processing time of all jobs in each machine. The corresponding data can be found according to the instance number.

To evaluate the quality of different algorithms, we selected randomly different cases and ran the two algorithms 10 times respectively to obtain the average value.There were many instances from Taillard's datasets, 4 of each of the sizes(jobs x machines) 20x5, 20x10, 20x15, 20x20.We implemented the Qlearning algorithm and our algorithm in python, running on a device with I7 CPU and 16 GB RAM.

In order to make the experiment more reasonable, we set the Qlearning algorithm to have the same episodes (max_episodes=10,000),learning rate α ($\alpha = 0.1$)and discount factor γ ($\gamma = 0.8$)as our algorithm to ensure that both algorithms operate under the same conditions. For the TS_Qlearning algorithm, we set the length of the tabu list to one-third of the number of jobs. The initial solution of the TS_Qlearning algorithm is obtained from the tabu list.We also compare the traditional algorithm GA, GA is an effective traditional flow shop scheduling algorithm.In the genetic algorithm, we use the following parameters: crossover_rate=0.4,mutation_rate=0.1 and iteration=1,000. In the experiment, we obtain the final sequence from the algorithm training, and then calculate the C_{max} value according to the final sequence.

In the experiment, we performed the Qlearning algorithm,the TS_Qlearning algorithm and GA for each Taillard problem. After running 10 times, the average of the experimental results was recorded in the table.Table 1 shows the different of scheduling instances selected to estimate the performance of the Qlearning algorithm ,GA and TS_Qlearning algorithm. In this case, we selected instances that fit different complexity problems based on the number of jobs and machines. This table summarizes the results of all 16 Taillard instances. Therefore a total of 480 experiments were run.The experiment results are shown in Table 1.

In general,these results from Table 1 show that the average idle time obtained by TS_Qlearning algorithm is better than that obtained by the Qlearning algorithm in any dataset. For the value of C_{max} , the result we obtain in TS_Qlearning algorithm is superior to that of Qlearning algorithm. Therefore, our algorithm has more advantages than Qlearning algorithm in solving scheduling problems.

Compared with the traditional GA algorithm, the advantages of TS_Qlearning algorithm in idletime value and C_{max} value are obviously reduced, and the GA algorithm is even better in the instance ta001,ta006 and ta015.We found that under this kind of datasets, learning to find the best job sequence does not fully take advantage of the RL algorithm.And In high-performance computing applications, users' requirements are often changeable,which will cause data to change.Compared with the traditional flow shop scheduling algorithm, RL method is able to learn from different environments, which may make it easier to meet the needs of users.

V. CONCLUSIONS AND FUTURE WORK

This paper presented TS_Qlearning algorithm to solve FSSP.From the results, we can conclude that our algorithm has more advantages than Qlearning algorithm in solving FSSP problems.For traditional scheduling algorithms such as genetic algorithm, the advantages of TS_Qlearning algorithm are not as great as expected.Perhaps training the two algorithms in the dynamic training data set can reflect the difference between the two algorithms.In the future, we will focus on solving more complex FSSP with reinforcement learning algorithm.It will be interesting to combine

TABLE I
QLearning algorithm,GA and TS_Qlearning results for the experiment

		Q-learning		**GA**		**Our Method**	
	Instance	**Idle time**	**Cmax**	**Idle time**	**Cmax**	**Idle time**	**Cmax**
20*5	ta001	941	1377.6	920.4	1367.3	922.4	1380.8
	ta003	767.6	1231.3	732.3	1205.1	**717.3**	**1201.3**
	ta005	988.8	1334.6	990.4	1336.8	985.3	1332.6
	ta006	664.4	1317.5	649.7	1304.1	658.2	1311
		Q-learning		**GA**		**Our Method**	
	Instance	**Idle time**	**Cmax**	**Idle time**	**Cmax**	**Idle time**	**Cmax**
20*10	ta011	3973	1857.9	3935.9	1935.7	**3910.2**	**1825**
	ta012	3905.2	1924.8	3815.1	1899.3	3804.6	1888.9
	ta015	2772.7	1665.5	2725.0	1657.8	2730.3	1662
	ta016	3760.8	1623.1	3706.8	1605.1	3658	1616.5
		Q-learning		**GA**		**Our Method**	
	Instance	**Idle time**	**Cmax**	**Idle time**	**Cmax**	**Idle time**	**Cmax**
20*15	ta03	10190.3	2176.7	10146.8	2171.3	10070.6	2165.4
	ta06	8757.1	2196.1	8575.2	2203.3	8521.8	2178.7
	ta08	8537.6	2171.7	8521.5	2131.8	**8375.1**	**2141.8**
	ta09	8239.6	2122.8	9369.1	2160.4	9145.9	2116.2
		Q-learning		**GA**		**Our Method**	
	Instance	**Idle time**	**Cmax**	**Idle time**	**Cmax**	**Idle time**	**Cmax**
20*20	ta022	14242.7	2385.2	14227.2	2396.4	14118.4	2357.4
	ta023	15829.9	2624.3	15905.7	5605.4	**15713.2**	**2587**
	ta025	17360.3	2586.8	17340.5	2654.5	17203.3	2580.2
	ta027	14929.6	2527.4	15988.9	2522	14802.1	2520.3

reinforcement learning algorithms with different scheduling rules (compound scheduling rules) and get better results just by using the learning algorithm.

Acknowledgment

This work was supported in part by National Key R&D Program of China(No. 2019YFB1707004).

References

[1] Orgerie A C, Assuncao M D, Lefevre L. A survey on techniques for improving the energy efficiency of large-scale distributed systems[J]. ACM Computing Surveys (CSUR), 2014, 46(4): 1-31.

[2] Čičková, Z.,Števo, S. (2010). Flow shop scheduling using differential evolution. Management Information Systems, 5(2), 008-013.

[3] Sutton, R. S., Barto, A. G. (2011). Reinforcement learning: An introduction.

[4] Leslie, D. S., Collins, E. J. (2005). Individual Q-learning in normal form games. SIAM Journal on Control and Optimization, 44(2), 495-514.

[5] Sayadi, M., Ramezanian, R., Ghaffari-Nasab, N. (2010). A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems. International Journal of Industrial Engineering Computations 1(1), 1-10.

[6] Fonseca-Reyna, Y. C., Martínez-Jiménez, Y.,Nowé, A. (2018). Q-learning algorithm performance for m-machine, n-jobs flow shop scheduling problems to minimize makespan. Investigación Operacional, 38(3), 281-290.

[7] Xiaokang Wang, Laurence T. Yang, Xingyu Chen, and Jian-Jun Han. A Tensor Computation and Optimization Model for Cyber-Physical-Social Big Data. IEEE Transactions on Sustainable Computing, vol. 4, no. 4, pp. 326-339,1 Oct.-Dec. 2019, doi: 10.1109/TSUSC.2017.2777503.

[8] Xiaokang Wang, Laurence T. Yang, Liwen Song, Huihui Wang, Lei Ren, M. Jamal Deen, “A Tensor-based Multi-Attributes Visual Feature Recognition Method for Industrial Intelligence,” IEEE Transactions on Industrial Informatics, DOI: 10.1109/TII.2020.2999901, 2020.

[9] X Wang, Y. Han, C. Wang, Q. Zhao, X. Chen and M. Chen. In-Edge AI: Intelligentizing Mobile Edge Computing, Caching and Communication by Federated Learning. IEEE Network, vol. 33, no. 5, pp. 156-165, Sept.-Oct. 2019, doi: 10.1109/MNET.2019.1800286.

[10] Sayadi, M., Ramezanian, R., Ghaffari-Nasab, N. (2010). A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems. International Journal of Industrial Engineering Computations 1(1), 1-10.

[11] L. Nai, R. Hadidi, J. Sim, H. Kim, P. Kumar and H. Kim .Graph-PIM: Enabling Instruction-Level PIM Offloading in Graph Computing Frameworks. IEEE International Symposium on High Performance Computer Architecture (HPCA), Austin, TX, 2017, pp. 457-468, doi: 10.1109/HPCA.2017.54.

[12] Hu Zhu, Tao Xie, Yusheng Guan, Lizhen Deng, Xiaokang Wang.Hypergraph Matching With an Entropy Barrier Function .IEEE Access, vol. 7, pp. 16638-16647, 2019, doi: 10.1109/ACCESS.2019.2895809.

[13] Cai Z, Duan Z, Li W. Exploiting Multi-Dimensional Task Diversity in Distributed Auctions for Mobile Crowdsensing[J]. IEEE Transactions on Mobile Computing, 2020.

[14] Zhu T, Shi T, Li J, et al. Task scheduling in deadline-aware mobile edge computing systems[J]. IEEE Internet of Things Journal, 2018, 6(3): 4854-4866.

[15] Lei Ren, Yuanjun Laili, Xiang Li, and Xiaokang Wang.Coding-based Large-Scale Task Assignment for Industrial Edge Intelligence.IEEE Transactions on Network Science and Engineering, doi: 10.1109/TNSE.2019.2942042.

[16] Liang Y, Cai Z, Yu J, et al. Deep learning based inference of private information using embedded sensors in smart devices[J]. IEEE Network, 2018, 32(4): 8-14.

[17] Ren L, Meng Z, Wang X, et al. A Wide-Deep-Sequence Model-Based Quality Prediction Method in Industrial Process Analysis[J]. IEEE Transactions on Neural Networks and Learning Systems, 2020, 31(9): 3721-3731.

[18] Stefan, P. É. T. E. R. (2003). Flow-shop scheduling based on reinforcement learning algorithm. Production Systems and Information Engineering, 1(1), 83-90.

[19] Henneberg, M., Neufeld, J. S. (2016). A constructive algorithm and a simulated annealing approach for solving flowshop problems with missing operations. International Journal of Production Research, 54(12), 3534-3550.

[20] Duan Z, Li W, Cai Z. Distributed auctions for task assignment and scheduling in mobile crowdsensing systems[C]//2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2017: 635-644.

[21] Fang X, et al. Job Scheduling to Minimize Total Completion Time on

Multiple Edge Servers[J]. IEEE Transactions on Network Science and Engineering, 2019.

[22] Liu, Y., Yin, M., Gu, W. (2014). An effective differential evolution algorithm for permutation flow shop scheduling problem. Applied Mathematics and Computation, 248, 143-159.

[23] Z. Cai and X. Zheng, ”A Private and Efficient Mechanism for Data Uploading in Smart Cyber-Physical Systems,” in IEEE Transactions on Network Science and Engineering, vol. 7, no. 2, pp. 766-775, 1 April-June 2020, doi: 10.1109/TNSE.2018.2830307.

[24] Miloš Šeda. Mathematical Models of Flow Shop and Job Shop Scheduling Problems[J]. Proceedings of World Academy of Science Engineering & Technolog, 2007(4).

[25] Mozer S, M C, Hasselmo M. Reinforcement Learning: An Introduction[J]. Machine Learning, 1992, 8(3-4):225-227.

[26] Sutton, R. S., Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

[27] Watkins, C. J., Dayan, P. (1992). Q-learning. Machine learning, 8(3-4), 279-292.

[28] G. Tesauro. Extending q-learning to general adaptive multi-agent systems. In Advances in neural information processing systems, pages 871–878, 2004.

[29] Jiménez, Y. M. (2012). A generic multi-agent reinforcement learning approach for scheduling problems. PhD, Vrije Universiteit Brussel, 128.

[30] Pezzella, F., Morganti, G., Ciaschetti, G.: A genetic algorithm for the flexible jobshop scheduling problem. Computers & Operations Research 35, 3202–3212 (2008)

[31] Lining, X., Chen, Y.: A Knowledge-Based Ant Colony Optimization for Flexible Job Shop Scheduling Problems. Applied Soft Computing (2009)

[32] Brandimarte, P.: Routing and scheduling in a flexible job shop by tabu search. Annals of Operations Research 41, 157–183 (1993).

[33] Martínez Y, Nowé A, Suárez J, et al. A reinforcement learning approach for the flexible job shop scheduling problem[C].International Conference on Learning and Intelligent Optimization. Springer, Berlin, Heidelberg, 2011: 253-262.

[34] BEASLEY,J.E.(1990): OR-Library.Disponible Consulted January 14, 2014. http://people.brunel.ac.uk/ mastjjb/jeb/info.html.

[35] Taillard, E. (1993). Benchmarks for basic scheduling problems. european journal of operational research, 64(2), 278-285.