A Reinforcement Learning-based Approach to Dynamic Job-shop Scheduling[1)]

WEI Ying-Zi[1,2] ZHAO Ming-Yang[1]
[1](*Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang* 110016)
[2](*Shenyang Ligong University, Shenyang* 110168)
(E-mail: wings_syit@126.com)

Abstract Production scheduling is critical to manufacturing system. Dispatching rules are usually applied dynamically to schedule the job in a dynamic job-shop. Existing scheduling approaches seldom address machine selection in the scheduling process. Composite rules, considering both machine selection and job selection, are proposed in this paper. The dynamic system is trained to enhance its learning and adaptive capability by a reinforcement learning (RL) algorithm. We define the conception of pressure to describe the system feature. Designing a reward function should be guided by the scheduling goal to accurately record the learning progress. Competitive results with the RL-based approach show that it can be used as real-time scheduling technology.

Key words Reinforcement learning, composite rules, mean tardiness, dynamic job-shop scheduling

1 Introduction

Scheduling problems essentially involve completing a set of jobs with a limited number of manufacturing resources under a number of constraints to optimize a particular objective function. These problems are known to be hard and usually belong to the NP-complete class of problems[1]. For the case of an actual shop floor, uncertainties (*i.e.*, machine breakdowns, material or tool shortages, transportation delays, *etc.*) complicate the scheduling problem, making it more difficult to solve. Thus, a dynamic scheduling system is more suitable to production application than the static one. The main difference of the dynamic and static scheduling algorithms is that the dynamic one requires more robustness and rapid reactivity to the changing environment.

Dispatching rules (DRs) are the most common approach in the dynamic scheduling system[1~5]. Use of dispatching rules is attractive because of their simplicity, low computation complexity and ease of implementation. A dispatching rule is concerned with selecting a job to be processed based on some criteria. It can realize the close-loop controlling of production process with one criterion. Rule-based scheduling is of the typical dynamic controlling scheme.

Enhancing the learning and adaptive capability of scheduling system has not received much attention. Although the dispatching rules do not guarantee an optimal schedule, they usually provide a reasonably good schedule. For the scheduler, making good decisions will significantly help to improve the scheduling performance. To use DRs appropriately for sequencing jobs, dynamic rule selection is required since the manufacturing shop status may change over time. A knowledge-based rule selection system can be used to rapidly respond to the changes of the shop status. However, the existing knowledge-based systems have the shortcoming that knowledge is acquired based on the use of off-line machine learning techniques. In addition, every resource selects the rules based on the same knowledge bases at the same period of time. Each resource unit should have its own knowledge base for DR selection.

Zhang *et al.*[6] applied $TD(\lambda)$ to a job-shop scheduling problem. The scheduling approach was an iterative repair-based scheduling method that started with generating a critical complete schedule by ignoring the resource constraints and incrementally repairing the schedule to find a shortest conflict-free schedule. Aydin *et al.*[7] proposed an intelligent agent-based scheduling system. They employed Q-III to train the agents to dynamically select dispatching rules. Their state determination criteria consist

1) Supported by 973 program of P. R. China (2002CB312200)
Received August 12, 2004; in revised form June 26, 2005

of the mean slack time of the queue and the buffer size of the machine. They take advantage of domain knowledge and experience in the learning process. Therefore, the principal learning mechanism of RL algorithm is omitted. Wang *et al.*[8] applied Q-learning to single machine job scheduling problem and provided recommendations for factor settings of RL. The literature review indicates that there has been little work on creating intelligent scheduling systems with a learning ability actually based on trial and error. In this paper, reinforcement learning is adopted for the dynamic scheduler to improve the on-line learning and adaptive capability.

2 Model of dynamic job-shop scheduling problem

The job-shop scheduling problem (JSP) is a general scheduling type that may be described as follows: given n jobs, each composed of several operations that must be processed on m machines. Each operation uses one of the m machines for a period of duration. Each machine can process at most one operation at a time and once an operation initiates processing on a given machine it must complete processing on that machine without interruption.

The processing times and inter-arrival times are defined according to a uniform distribution varying between G_1 and G_2. The due date DD_i of job i with release time g_i and processing time p_{ij} is determined as

$$DD_i = g_i + k\sum_{j=1}^{m} p_{ij}, \quad k \in [-1, 4] \tag{1}$$

where DD_i, g_i, m, k and p_{ij} represent due date, arrival time, the number of operations, coefficient of tightness, and processing time of ith operation, respectively. Since k can take on negative values, we may have jobs that are already tardy when they become available. This is often the case in industrial situations where a job may be delayed in preceding stages of the manufacturing process.

The mean tardiness of finished jobs for the scheduling system is $\bar{T} = \frac{1}{n}\sum_{i=1}^{n} T_i$. Mean tardiness ($\bar{T}$) is selected as the performance criterion for measuring the efficiency and effectiveness of our scheduling system. The tardiness of job i is $T_i = \max\{C_i - DD_i, 0\}$. The lateness of job i is defined as $L_i = C_i - DD_i$, where C_i is the completed time of job i, DD_i represents the due date of job i. Tardiness is the positive part of lateness, so the lateness is used to predict the scheduling performance when scheduling is in the preceding stage. To describe the intermediate feature of scheduling system, we introduce two new definitions to determine status of the scheduling system. At each scheduling moment, the two definitions are used as follows.

Definition 1. The estimated mean remaining processing time (EMRT) is defined as

$$EMRT = \frac{1}{n}\sum_{i}^{n}\sum_{j=1}^{u} p_{ij} \tag{2}$$

Definition 2. Each time when the resource of scheduling system changes is considered as a scheduling moment.

Definition 3. The estimated mean lateness (EMLT) is defined as

$$EMLT = \frac{1}{n}\sum_{i=1}^{n}\left[\sum_{j=1}^{\mu} p_{ij} - (DD_i - schedule\_time_i)\right] \tag{3}$$

In (2) and (3), u is the number of unfinished operations of job i and $schedule\_time_i$ is the current scheduling time of job i.

3 Constructing new composite rules

Existing scheduling rules seldom address machine selection[5~9]. The often occurrence is that machine selection is based on random or on some given sequence. However, dynamic events, such as random job arrivals, machine breakdown/repair, and different cost of finishing the same operation with different machines, make machine selection critical for scheduling system in the case of heavy loading and tight due date limitation[10]. These motivate us to use the dispatching rule for machine selection.

Three new two-step scheduling rules, named as composite scheduling rules, are proposed. We call them rule 0, rule 1 and rule 2. Rule 0: at first step, the scheduling system selects the job with the smallest CR (critical ratio) value (*i.e.*, according to CR rule). Then the job is entailed to the machine with the earliest finish time (EFT) to finish its operation on this machine. Rule 1: the scheduler first selects a job which first comes to the scheduling system, then the machine which may finish its operation with EFT has the highest priority rating for selection. Rule 2: the machine with the earliest available time is selected at first. This strategy is mainly to reduce machine′s idle time and improve the scheduling performance. At the second step, a job with EFT on the selected machine is scheduled among the available jobs.

At each scheduling moment, a composite rule is selected out for scheduling the job and machine. If it is assumed that the job-shop problem has m machines and n jobs, then the dimension of a complete schedule represented by the rule set will be $m * n$.

4 Q-learning application to composite rule selection

RL is on-line actor critic method in machine learning[11,12]. The interaction of a learning system with the environment is its major source of intelligence. SARSA and Q-learning have been adopted widely to optimize the learning system for they are model free. Most RL algorithms iteratively improve estimates of value functions based on samples of transitions obtained on-line. For example, at each time step t, the typical tabular learning algorithm updates the value of the current state-action pair (s_t, a_t) based on the observed reward r and the next state-action pair (s', a'), as

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r + \gamma \max_{a' \in A} Q(s', a')) \tag{4}$$

where α is the learning rate, γ is the discount coefficient.

Reinforcement learning is a dynamic programming approach for the dynamic scheduling problem of discrete events. According to the principle of dynamic programming, the key to realize an effective learning is how to formulate a scheduling problem into an RL problem. The sub-problems include: setting state determination criterion, constructing the policy table, developing the reward function and applying an effective strategy for the state search.

4.1 State determination criteria

The key to application of dynamic programming is to identify the states character of multi-stage decision. Proper choice of state variable is essential for the dynamic programming problem. State variable should express accurately the current situation of the system and different stages should have different state description. So, in the Q-learning algorithm, we define a novel state variable to describe the state of dynamic scheduling environment.

When the system is under heavy loading conditions and jobs are assigned with very tight due dates, most of the jobs will be tardy. The estimated mean lateness (EMLT) grows gradually as scheduling actions are executed one by one. Contradictorily, the estimated mean remained processing time (EMRT) becomes smaller and smaller. The scheduling system has much pressure to finish the processing task as fast as possible. We present a novel state determination criterion in which state is defined based on the concept of pressure. Pressure, denoted by the ratio of EMLT and EMRT, is given as follows

$$pressure = \frac{\text{EMLT}}{\text{EMRT} + \delta} \tag{5}$$

We calculate the pressure by dividing EMLT by EMRT and δ. In (5), δ is a constant in order to avoid being divided by 0.

Each state of different stage is defined uniquely by this state variable. The concept of pressure may accurately describe the current situation at each scheduling moment. The only one variable, pressure, provides two variables′ information (*i.e.*, EMLT and EMRT). Another advantage of using this description of state is that this strategy may save the memory of computation for Q-value table. Table 1 provides an example of a policy table with 6 states determined by pressure.

In Table 1, d is the adjustable coefficient that makes the continuous state being partitioned into states. Using few number of states may cause the scheduler unable to differentiate between the decisions since they lie within the same range. The use of more ranges in the reward function permits the reward or penalty associated with each decision to be expressed more precisely. However, using too much number of states will lead search to slow convergence into optimum.

Table 1 An example of a 6-state policy table

State	State determination criteria	Rule 0	Rule 1	Rule 2
0	if(Pressure<0) $s = 0$;	$Q(0,0)$	$Q(0,1)$	$Q(0,2)$
1	if(Pressure>=0&&Pressure<d) $s = 1$;	$Q(1,0)$	$Q(1,1)$	$Q(1,2)$
2	if(Pressure>= d&&Pressure< 2^*d) $s = 2$;	$Q(2,0)$	$Q(2,1)$	$Q(2,2)$
3	if(Pressure>= 2^*d&&Pressure< 3^*d) $s = 3$;	$Q(3,0)$	$Q(3,1)$	$Q(3,2)$
4	if(Pressure>= 3^*d&&Pressure< 4^*d) $s = 4$;	$Q(4,0)$	$Q(4,1)$	$Q(4,2)$
5	if(Pressure>= 4^*d) $s = 5$;	$Q(5,0)$	$Q(5,1)$	$Q(5,2)$

4.2 Developing the reward function

The reward function using the grading method is widely used in the literature.

$$r_t = \begin{cases} 1 & \textit{good states} \\ -1 & \textit{bad states} \\ 0 & \textit{other states} \end{cases} \tag{6}$$

Mean tardiness of finished jobs can be calculated only after scheduling finishs. If this grading method of formula (6) is adopted as the reward function for evaluating states, the reward value will be null during the intermediate states of the scheduling process. This will lead search to a random procedure for the nonterminal states of scheduling.

Designing a reward function should be guided by the goal of the learning system. In this study, the scheduling objective is to minimize the mean tardiness of finished jobs. Therefore, jobs′ EMLT is used to determine the amount of reward or penalty for the scheduler′s decision (composite rules selection). The larger EMLT is, the greater the penalties assigned to the learning system. When the EMLT value is negative, it is predicable that the scheduling system will finish the production task without delay. Then the learning system is rewarded. Therefore, jobs′ EMLT is used for designing the reward function as follows.

$$r_t = e - q \cdot \mathrm{EMLT} \tag{7}$$

where e and q are constants of positive value for regulating EMLT to the reward received by the learning system. The formula (7) turns the minimum problem for EMLT into the process of maximizing reward. This reward function helps to differentiate the performance of different actions in the same state.

4.3 Action selection policy

There are two types of strategies to select actions. One is called exploration that an action is determined randomly to try various actions and state transitions. The other is called exploitation that for each possible action its next state is searched, and the action is selected which produces the maximum sum of the reward and the value of the next state, after learning.

The ε-greedy method is adopted in our study. The ε-greedy action selection is used because of its simplicity and it often ensures a sufficient exploitation/exploration balance. A policy is called greedy with respect to some action value function $Q(s,a)$ if in each state it selects one of the actions that has the maximum value:

$$\pi_t(s,a) = \arg\max_{a' \in A} Q(s,a') \tag{8}$$

4.4 Searching stop condition

There are two different factors that determine the utility of an action. These are the immediate reward, and the action value of the state to which a transition occurs as a result of that action. When a system visits a state, an action with the highest (or lowest for minimization) action value is chosen (*i.e.*, using greedy policy). Initially, the action values for all state-action pairs are assigned arbitrary equal values (usually zeros). When visiting a state for the first time, and several other times during the initial learning phase, the learning system explores the environment by taking random actions. As the system revisits the state, the Q-learning algorithm selects the action based on the current action values. If a good state is brought upon the action of certain rule, the Q-learning will reinforce the rule for this state. As good actions are rewarded and bad actions are punished over time, for every state, the action values of a smaller subset (one or more) of the actions tend to grow and other diminish. The learning phase ends when a clear trend appears with one or more actions in every state being dominant.

5 Experimental results

Compared with the simulation example provided by Aydin *et al.*[7], our scheduling problem has more calculation complexity than theirs. The problem is to organize the execution of 15 jobs on 9 different machines. The processing time and inter-arrival time are the number distributed uniformly over the interval[2,9].

Table 2 lists the comparison result of rule-based scheduling performance. It shows that significant improvement to the scheduling performance can be achieved through the use of simple machine selection rules. The curve of Q-learning performance fluctuates very large in Fig. 1. For combinatorial optimization problems, even one bad explore step will affect the results heavily. However, the tendency of performance curve is descending, which results from the search balance of explore and exploit. The fluctuation is helpful for the learning system to find better results. In Table 3, the data are the average values obtained by the program running 10 times. The performance of the Q-learning scheduler was tested with respect to various values of k. At the end of training, the Q-learning approach gave better results than the aforementioned alternatives. The only one item over which conventional rules have superiority is computation time. However, in our experience of cooperating with real-world manufacturers, one second and ten seconds actually do not make huge difference to them, for them both satisfy the requirement of real dynamic scheduling. Action selection based on the ε-greedy strategy was implemented, in which ε is gradually changed from a big value to a small one during the learning process. When ε equals 0, the search process will converge fast to an optimal solution. This shows the robustness and convergence of RL. Fig. 2 illustrates a scheduling Gantt diagram which shows the high machine occupy rate and compact working procedure.

Table 2 Comparison of rule-based scheduling performance

	CR	FCFS	EFT
Dispatching rule	86.43	50.47	42.56
Composite rule	30.87	32.33	28.73

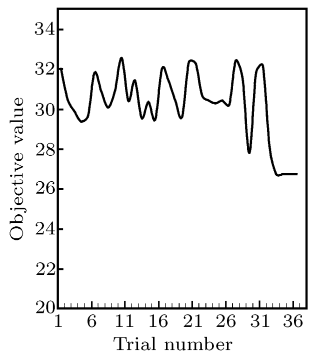

Fig. 1 Performance curve of Q-learning scheduling approach

Table 3 Experimental results of dynamic job-shop problem with 9 jobs and 15 machines

Rule List	k			
	$k = 1$	$k = 1.2$	$k = 1.5$	$k = 2$
Rule 0	30.87	23.73	9.13	0
Rule 1	32.33	26.87	11.07	1.2
Rule 2	28.73	23.27	8.47	0
Rules random combined	31.13	24.82	11.55	0.17
Q learning	28.20	22.24	8.67	0

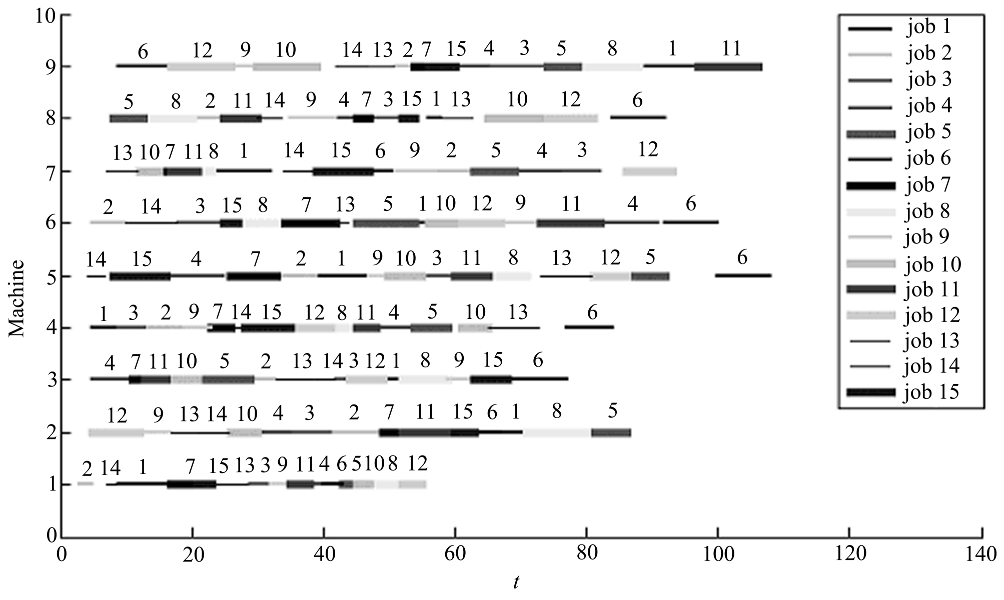

Fig. 2 Scheduling Gantt diagram of 15 jobs and 9 machines

According to the simulation experiment using Borland C++ DOS programming environment on the PentiumIV1.5GHz and RAM128MHz computer, a satisfying solution will be obtained in 12 seconds through Q learning approach. This indicates that methods we present here have the high effectiveness and efficiency to solve dynamic scheduling problem.

6 Conclusions and future research

Due to complexity in the manufacturing system, most applicable scheduling system is based on rules scheduling. In this paper, we present an iterative optimization framework for dynamic scheduling system using reinforcement learning. This study not only investigates the main effects of reinforcement learning application, but also extends RL algorithm to the field of production scheduling. Our main idea is to define an intermediate state variable to describe the whole scheduling process. This idea is original for RL system to break the limitation that RL only evaluates terminal states of scheduling system. Q-learning algorithm is model free, so it is suitable for most problems for searching optimal states whose states transition depends on a scheduler. The Q-learning based approach seems promising for developing a versatile on-line learning scheduler in future.

As the scheduling objective is to minimize mean tardiness in this research, the state determination criterion and reward function are both built based on mean tardiness. If the goal of Q-learning scheduling changes, the state determination criterion and reward function should be modified accordingly. For Q-learning algorithm, greedy action selection will lead searching to a fast convergence. But the optimal solution may be a local optimum. Future research will address the global optimization for

RL algorithm based on the mechanism of parallel search. An alternative approach is to combine the ability of on-line local optimization by RL with the asynchronous global combinational optimization by genetic algorithm.

References

1 Xu Jun-Gang, Dai Zhong-Guo, Wang Hong-An. An overview of theories and methods of production scheduling. *Journal of Computer Research and Development*, 2004, **41**(2): 257∼267

2 Sun D, Lin L. A dynamic job shop scheduling framework: A backward approach. *International Journal of Production Research*, 1994, **32**(4): 967∼985

3 Mohanasundaram K M, Natarajan K, Viswanathkumar G, Radhakrishnan P, Rajendran C. Scheduling rules for dynamic shops that manufacture multi-level jobs. *Computers & Industrial Engineering*, 2003, **44**(1): 119∼131

4 Sun Rong-Lei, Xiong You-Lun, Du Run-Sheng, Ding Han. Iterative optimization of rule-based scheduling. *Computer Integrated Manufacturing System*, 2002, **8**(7): 546∼550

5 Bowden R O, Bullington S F. Development of manufacturing control strategies using unsupervised machine learning. *IIE Transactions*, 1996, **28**(4): 319∼331

6 Zhang Wei. Reinforcement Learning for Job-Shop Scheduling. [Ph. D. Dissertation], Oregon State University, 1996

7 Aydin M E, Oztemel E. Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems*, 2000, **33**(2): 169∼178

8 Wang Yi-Chi. Application of reinforcement learning to multi-agent production scheduling, [Ph. D. Dissertation], Mississippi State University, 2003

9 Pinedo M. Scheduling Theory, Algorithms, and Systems. Prentice Hall. 1995

10 Subramaniam V, Lee G K, Ramesh T, Hong G S, Wong Y S. Machine selection rules in a dynamic job-shop. *The International Journal of Advanced Manufacturing Technology*, 2000, **16**: 902∼908 Springer

11 Sutton R S, A Barto G. Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 1998

12 Gao Yang, Chen Shi-Fu, Lu Xin, Research on reinforcement learning technology: A review. *Acta Automatica Sinica*, 2004, **30**(1): 86∼100

WEI Ying-Zi Received her master and Ph. D. degrees from Liaoning Institute of Technology in 1999 and Shenyang Institute of Automation, Chinese Academy Sciences in 2005, respectively. Her research interests include machine learning theory, the theory of manufacturing system and its application.

ZHAO Ming-Yang Received his Ph. D. degree from Northeastern University in 1995. His research interests include robotics and advanced manufacturing technology.