Int J Adv Manuf Technol (2009) 42:312–320
DOI 10.1007/s00170-008-1588-0

ORIGINAL ARTICLE

An adaptive approach to dynamic scheduling in knowledgeable manufacturing cell

H.-B. Yang · H.-S. Yan

Received: 10 August 2007 / Accepted: 29 May 2008 / Published online: 28 June 2008

Abstract To overcome deficiency in the global capacity of a single dispatching rule, it is vital to select a dispatching rule in real time for dynamic scheduling. Among the studies addressing the method for selecting dispatching rules, few have no requirements for domain knowledge or accurate training example, which is hard to acquire from the real production system. In this paper, a new learning algorithm, along with the presentation of an adaptive scheduling control policy, is proposed to obtain the dynamic scheduling knowledge effectively, and different dispatching rules are selected to schedule the jobs in the machine buffer according to the current transient state of the system. Case studies are given to illustrate the validity of the scheduling control policy.

Keywords Control policy · Dispatching rules · Dynamic scheduling · Knowledgeable manufacturing cell

1 Introduction

With the rapid development in the technologies of electrons, information, artificial intelligence, and control, various advanced manufacturing modes and manufacturing conceptions, such as intelligent manufacturing [1], lean production [2], green manufacturing [3], agile manufacturing [4, 5], and concurrent engineering [6, 7], have emerged. Most of them, capable of meeting certain demands of the manufacturing system, still have limitations of their application scope due to the multiple demands of various enterprises. Therefore, knowledgeable manufacturing, a new manufacturing idea brought forward in 2000 [8], is being given more and more attention. The technique takes an advanced manufacturing mode as advanced manufacturing knowledge, so that all kinds of complementary advanced manufacturing modes can be transformed into their corresponding advanced manufacturing knowledge in the advanced manufacturing system. On the basis of one-to-one isomorphic mapping between agent mesh and knowledgeable mesh, the existing advanced manufacturing mode can be incorporated into the knowledgeable manufacturing system (KMS) to meet different demands from different enterprises [9–11]. The KMS, a highly intelligent manufacturing system characterized by self-adaptation, self-learning, self-evolution, self-reconfiguration, self-training, and self-maintenance, can constantly adapt to environmental change by self-learning and self-evolution.

Self-adaptation, as a key technology in the realization of KMS, involves many fields in the manufacturing system. In real production, dynamic scheduling often faces manufacturing situation with many unpredicted turbulences, like stochastic arrival or uncertain process time of job, uncertain delivery date, machine failure, and shortage of raw material, which raise many scheduling problems as dynamic scheduling to be dealt. At present, the research on dynamic scheduling mainly focuses on optimization [12], systems simulation [13], heuristic algorithm [14, 15], multi-agent method [16, 17], and artificial intelligence method [18–20].

H.-B. Yang · H.-S. Yan (✉)
Key Laboratory of Measurement and Control of Complex Systems of Engineering, Ministry of Education, Southeast University,
Nanjing, Jiangsu 210096, China
e-mail: hsyan@seu.edu.cn

To the authors' best knowledge, most of the dynamic scheduling problems prove to be NP-hard. The dispatching rule method, as a heuristic algorithm, bearing advantages of insensitivity to the NP characteristic and real-time quality, is widely used in practice.

Under different system parameter conditions and by simulation experiments and making comparisons of various dispatching rules, Baker [21] found crossover points between performance curves of different dispatching rules. They concluded that there was no dominant optimal dispatching rule for different scheduling criteria; i.e., when there was a change in the manufacturing system state, former effective dispatching rule may become less effective. As a result, a single dispatching rule may have a poorer overall effect, and for a dynamic scheduling with frequent changes of the system parameter, it does not provide the ideal effect.

In order to improve the capacity of dispatching rule, Arzi and Iaroslavitz [22] used a neural network to construct a flexible manufacturing cell controller, which can train the neural network to choose the proper dispatching rules. Nevertheless, this method requires a long training period with a poor explanation capacity, which also complicates the network structure as the problem scale grows. Wan [23] used the nearest neighbor method to train fuzzy system under job shop environment, so the scheduling rules are selected dynamically in terms of training results. However, with few scheduling rules employed, expert knowledge is hard to gain.

Piramuthu et al. [24] presented an adaptive scheduling policy for dynamic manufacturing system scheduling using information obtained from snapshots of the system at various points in time. As noted by the authors, the training examples were driven by simulation experiments; the appropriateness of a dispatching rule for a given pattern is therefore determined by its steady-state average performance over the length of the simulation run. However, a dispatching rule may perform well in the long run for a given set of attributes; it need not to be effective when applied on a rolling basis on transient patterns. Due to the randomness and uncertainty of parameter changes in the manufacturing systems and the frequent disruptions, accurate training examples on dispatching rules are difficult to acquire, which also exists in the literature [22].

As seen from the above, there is a lack of research focused on avoiding the difficulty of training-example acquisition as mentioned above. In this paper, B–Q learning algorithm is proposed based on reinforcement learning and knowledgeable manufacturing cell (KMC). Then, an adaptive scheduling control policy (BQ_ASCP), adaptable to manufacturing environment change, is constructed by the algorithm. For simplicity, mean tardiness is used as the only criterion for carrying out the algorithm evaluation.

2 Problem formulation

The limited capacity of manufacturing facilities and uncertain factors, such as failure of raw material supply, etc., may prevent a company from finishing a product order on time. Delay on delivery brings not only tremendous economic loss to downstream enterprises but also the penalty from customers. Thus, minimizing the job tardiness is a deep concern of manufacturing enterprises and a major issue in production line scheduling. In this respect, Koulamas has given a general review of the single machine scheduling to minimize total tardiness [25].

KMS is a dynamic system composed of many KMCs, which include processing agents, transport agents, decision-scheduling modules, and testing devices like raster, sensors, and so on. Consequently, KMC has capabilities of real-time monitor, data acquisition, information processing, and decision making. Consider a KMC including M processing agents, each of which has a buffer, and N jobs to be scheduled to gain the minimized job mean tardiness. Each job consists of many manufacturing processes. Workpieces arrive stochastically at KMC, waiting for processing. Jobs are independent to one another and without priorities. A machine can only process one workpiece in a period of time, and the processing job cannot be interrupted or rearranged. Let d_j denote the ideal due date for the jth workpiece and C_j denote the actual completion time; then, the objective function of minimized mean tardiness can be defined as:

$$J = \min \left[\sum_{j=1}^{N} \varphi \max\left(C_j - d_j, 0\right) \Big/ N \right] \tag{1}$$

where φ is the punishment factor for workpiece tardiness.

Our research aims to select dynamically effective dispatching rules for scheduling the workpieces in processing agent buffer in a rational way, so as to minimize the mean tardiness of workpieces in KMC.

3 Analysis on B–Q learning algorithm

Reinforcement learning is an important machine learning method by evaluating the environment feedback and then improving the mapping of system state to action. It is different from supervised learning in connection doctrine. The signal of reward, provided by external surroundings, is only feedback evaluation on mapping between system state and action rather than directly tells the reinforcement learning system how to generate correct actions, as shown in its learning mechanism illustrated in Fig. 1. Reinforcement learning includes many algorithms, such as *TD* (λ) (temporal difference method), Q-learning algorithm, adap-

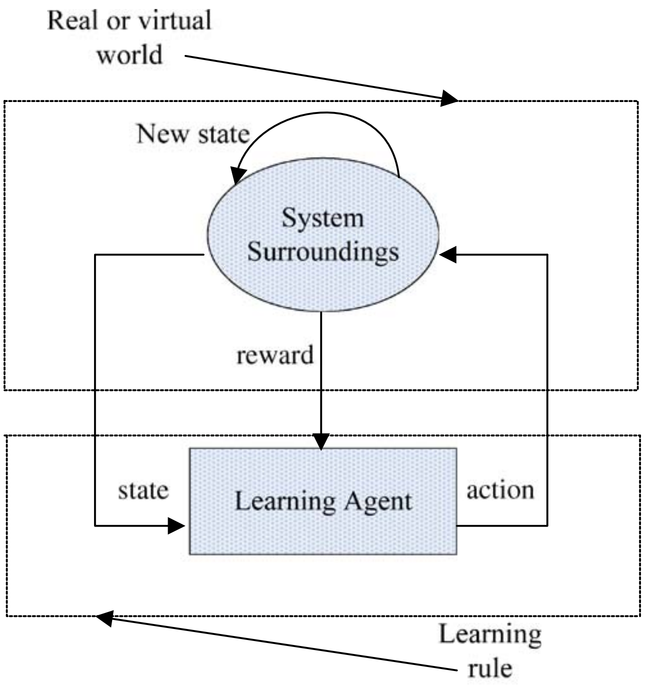

Fig. 1 Learning mechanism of reinforcement learning

tive heuristic critic algorithm, etc. The Q-learning algorithm, proposed by Watkins in 1989, is considered as one of asynchronous dynamic programming.

Currently, reinforcement learning technology has found wide application in manufacturing processes, including inventory control, supply chain management, machine reconfiguration, and preventive maintenances [26, 27]. However, little has been written about the application of reinforcement learning technology to dynamic scheduling in the existing literature. This is possibly caused by structural credit assignment of reinforcement learning, in which the learning results will be bad and cannot meet actual production requirements when facing large-scale state-space problems in dynamic scheduling. To eliminate the poor effect of a large state space, B–Q learning algorithm is proposed here to combine the basic sequential algorithmic scheme (BSAS) [28] with the Q-learning algorithm. The B–Q learning algorithm causes clustering in the state yielded by the KMC simulator and learns on the basis of lots of clustering states. Therefore, only a minor scale state space needs to be searched, so that its learning efficiency is improved and generalization capability is enhanced.

3.1 Selection of state features

Definition 1 At dispatching decision-making time, the sum of the rest processing times of all the jobs in the buffer of each processing agent in KMC is calculated, the maximum of which is called maximum machine workload, marked as $\omega_{\max}$.

Definition 2 At dispatching decision-making time, the sum of the rest processing times of all the jobs in the buffer of each processing agent in KMC is calculated, the mean value of which is defined as mean machine workload, marked as $\overline{\omega}$.

Definition 3 The ratio of KMC's maximum machine workload to the mean machine workload is defined as relative machine workload, described as ω, i.e., $\omega = \omega_{\max}/\overline{\omega}$.

Let S represent the state space of KMC; then $S=\cup s_i$, in which s_i consists of system state features and represents the actual state of KMC operation. At least more than ten of the state features are needed to describe a certain transient state of a production system entirely. Under such high-dimensional conditions, yielded state numbers increase exponentially, which causes state numbers to be extremely large if only several state features change. To reduce such impact, we select four state features—the mean allowance factor, system utilization, relative machine workloads, and average slack time, which mainly affect the dispatching rules performance. Thus, s_i is composed of four state features, that is $s_i = (\omega, f, \mu, \zeta)$, where feature variable ω is the relative machine workload. f is the mean allowance factor of the system and is defined herein as

$$f = \sum f_j \Big/ N_d \tag{2}$$

where f_j is the jth job allowance factor, reflecting whether the due date of the jth job is tight or loose, and N_d is the total number of jobs in the system.

Feature variable μ is system utilization rate, i.e., the ratio of the current non-idle processing agent number to the total processing agent number in KMC. State feature ζ represents mean slack time; if ζ_j is the slack time of the jth job, we have

$$\zeta_j = d_j - t - \sum_{q=k_d}^{k_j} p_{jq} \tag{3}$$

where t is the current time, P_{jq} the required time for operation q of the jth job (if operation q is under processing, then P_{jq} means the remaining processing time for this operation), k_d the operation under processing or waiting for processing, and k_j the total operation number for the jth job. Thus, the average slack time is defined herein as

$$\zeta = \left(\sum \zeta_j\right) \Big/ N_d \tag{4}$$

Definition 4 In the process of KMC running, if x clusters are obtained through BSAS clustering the system state, then the center of all the system states in the uth cluster is called clustering state s_u^c, so there are x clustering states, defined as $S^c = \cup s_u^c$, for $u = 1, 2, \cdots, x$.

The dissimilarity measure adopted in the BSAS is the Euclidean distance in terms of the feature value magnitude. To keep feature values within similar ranges, it is necessary to normalize feature values. Among all normalization preprocessing methods, the scale factor approach is relatively simple and useful. It is easy to divide a feature value that needs preprocessing by a simple scale factor. By selecting scale factor properly, one can balance the contribution of the respective feature value and retain the original semantics of the feature. Thus, in this paper, the scale factor method is chosen to make normalization preprocessing on the four state features and to balance each feature's function in state clustering as well.

3.2 Proposition of B–Q learning algorithm

At any scheduling decision-making point, KMC will select the correct action a_t according to the current system clustering state s_t^c namely, whose dispatching rule will be chosen to schedule the process-awaiting job. Action a_t is one of the actions in the set A, i.e., $a_t \in A$, and A is a set composed of many actions, that is, $A = \{a_1, a_2, \cdots, a_\beta\}$. Each action in A corresponds to a dispatching rule. A following system clustering state s_{t+1}^c is yielded after the action a_t acts on the KMC system. The learner thus receives an immediate reward r_{t+1}. Thus, a series of rewards will be yielded in the process of KMC running. In order to learn the optimal control policy according to all subsequent rewards on the time axis, that is, to select the optimum action by the detected clustering state, the evaluating function is defined below.

Definition 5 KMC starts from the system clustering state s_t^c at t time and selects action a_t according to a certain control policy, so that the expected cumulative discounted reward is obtained by complying with the policy, which is called evaluation function on the state-action pair (s_t^c, a_t), marked as $Q(s_t^c, a_t)$.

For any state-action pair (s_u^c, a_v), the expression $Q(s_u^c, a_v)(1 \le u \le x, 1 \le v \le \beta)$ can be obtained by the definition of evaluation function. Thus, we have

$$Q(s_u^c, a_v) = E\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t^c = s_u^c, a_t = a_v\right\} \tag{5}$$

where $\gamma (0 \le \gamma < 1)$ is a discount rate for delayed reward, reflecting the relative ratio of delayed reward to immediate reward.

Beginning with KMC state-action pair (s_u^c, a_v), each step selects an action a following the optimal control policy, so that the maximum expected cumulative reward is obtained, i.e., the optimal evaluation function $Q^*(s_u^c, a_v)$. The essence for optimal control policy learning is to learn how to get the optimal evaluation function. However, due to the clustering state in the evaluation function $Q(s_t^c, a_t)$, the value of evaluation function fluctuates violently, resulting in poor stability of the learning course. To reduce such fluctuation, the evaluation function threshold Θ is introduced to acquire the iteration learning model for B–Q learning algorithm, as shown in Eq. 6.

$$Q_n(s_t^c, a_t) = \begin{cases} Q_{n-1}(s_t^c, a_t) + \alpha_n(s_t^c, a_t)\left[r_{t+1} + \gamma \max\limits_a Q_{n-1}(s_{t+1}^c, a) - Q_{n-1}(s_t^c, a_t)\right], \\ \text{if} \quad \left|r_{t+1} + \gamma \max\limits_a Q_{n-1}(s_{t+1}^c, a) - Q_{n-1}(s_t^c, a_t)\right| \le \Theta \\ Q_{n-1}(s_t^c, a_t) + \alpha_n(s_t^c, a_t)\left[r_{t+1} + \gamma \max\limits_a Q_{n-1}(s_{t+1}^c, a) - Q_{n-1}(s_t^c, a_t) - \phi(n)\Delta_{\Theta 1}\right], \\ \text{if} \quad r_{t+1} + \gamma \max\limits_a Q_{n-1}(s_{t+1}^c, a) - Q_{n-1}(s_t^c, a_t) > \Theta \\ Q_{n-1}(s_t^c, a_t) + \alpha_n(s_t^c, a_t)\left[r_{t+1} + \gamma \max\limits_a Q_{n-1}(s_{t+1}^c, a) - Q_{n-1}(s_t^c, a_t) - \phi(n)\Delta_{\Theta 2}\right], \\ \text{if} \quad r_{t+1} + \gamma \max\limits_a Q_{n-1}(s_{t+1}^c, a) - Q_{n-1}(s_t^c, a_t) < -\Theta \end{cases} \tag{6}$$

where $\Delta_{\Theta 1}$ and $\Delta_{\Theta 2}$ are defined herein as

$$\Delta_{\Theta 1} = r_{t+1} + \gamma \max_a Q_{n-1}(s_{t+1}^c, a) - Q_{n-1}(s_t^c, a_t) - \Theta \tag{7}$$

$$\Delta_{\Theta 2} = r_{t+1} + \gamma \max_a Q_{n-1}(s_{t+1}^c, a) - Q_{n-1}(s_t^c, a_t) + \Theta \tag{8}$$

In Eq. 6, $Q_n(s_t^c, a_t)$ is the evaluation function of the nth cycle. α_n, the step parameter, is reduced gradually at a certain rate in the learning process, meeting the aim of converging to the optimal evaluation function. It can be derived by the following expression:

$$\alpha_n(s_t^c, a_t) = C_\alpha / (1 + \text{visits}_n(s_t^c, a_t)) \tag{9}$$

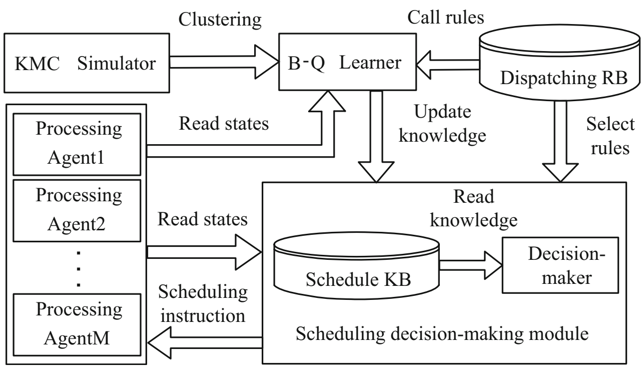

Fig. 2 Physical framework of adaptive control policy

where C_α is a weight coefficient variable of the step parameter, $\text{visits}_n\left(s_t^c, a_t\right)$ is the total visit time of state-action pair $\left(s_t^c, a_t\right)$ in n cycles. In consequence, the step parameter α_n diminishes when the visit time increases. In the course of learning, when KMC is at state s_t^c, the learner will select action a from the set A by adopting *ε-greedy* method, that is, the action a of the maximum evaluation function $\max\limits_{a_t} Q\left(s_t^c, a_t\right)$ is selected by probability $(1-\varepsilon)$; other actions in the set A are selected randomly by probability ε. Based on the above, the following provides brief details of the proposed B–Q learning algorithm:

Algorithm 1: Algorithm 1 is the B–Q learning algorithm.

Step 1 Initialize cluster number x=1 and i=1. Set maximum cluster number K, dissimilarity threshold Ω in BSAS, and the maximum number of states κ from KMC simulator.

Step 2 Perform KMC simulator, and B–Q learner acquires the initial state s_1 generated by simulator. Normalize the feature values in state s_1, then the xth cluster $C_x = \{s_1\}$.

Step 3 Set $i = i + 1$ and normalize the feature values in state $s_i (2 \leq i \leq \kappa)$. By calculating the dissimilarity measure $d(s_i, C_l)$ between the state s_i and the cluster $C_l (1 \leq l \leq x)$ with Euclidean distance, we obtain the cluster C_h with the following satisfied:

$$d(s_i, C_h) = \min_{1 \leq l \leq x} d(s_i, C_l) \tag{10}$$

Step 4 If $x < K$ and $d(s_i, C_h) > \Omega$, let $x = x + 1$ and $C_x = \{s_i\}$. Otherwise put state s_i into the cluster C_h, that is $C_h = C_h \cup s_i$, and recalculate the clustering state s_h^c; then, return to step 3. After the clustering of all κ states is completed, we obtain x cluster C_l and clustering state $s_u^c, l = 1, 2, \cdots, x$, $u = 1, 2, \cdots, x$.

Step 5 Initialize the evaluation function of all action-state pair $\left(s_u^c, a_v\right)$, referred as $Q_0\left(s_u^c, a_v\right)$, for $u = 1, 2, \cdots, x$, and $v = 1, 2, \cdots, \beta$. Set cycle time n=1. When KMC begins to operate, select action a_{t_0} randomly from set A.

Step 6 Set $n = n + 1$. B–Q learner detects the state s_t that KMC stays at time t and calculates dissimilarity measure $d(s_t, C_l)$, for $l = 1, 2, \cdots, x$. Then we have

$$d(s_t, C_u) = \min_{1 \leq l \leq x} d(s_t, C_l), \tag{11}$$

so the clustering state s_t^c is obtained at time t, i.e., state s_u^c. Accordingly, B–Q learner selects the action

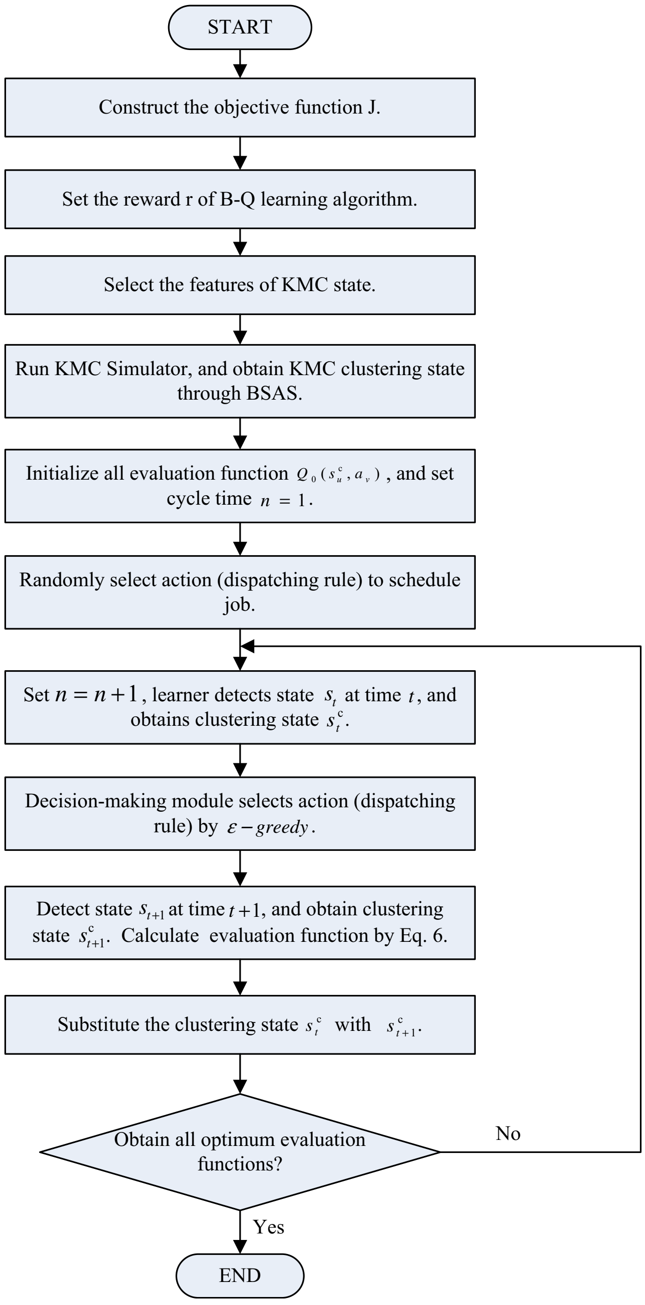
START
Construct the objective function J.
Set the reward r of B-Q learning algorithm.
Select the features of KMC state.
Run KMC Simulator, and obtain KMC clustering state through BSAS.
Initialize all evaluation function $Q_0(s_u^c, a_v)$, and set cycle time $n = 1$.
Randomly select action (dispatching rule) to schedule job.
Set $n = n+1$, learner detects state s_t at time t, and obtains clustering state s_t^c.
Decision-making module selects action (dispatching rule) by $\varepsilon-greedy$.
Detect state s_{t+1} at time $t+1$, and obtain clustering state s_{t+1}^c. Calculate evaluation function by Eq. 6.
Substitute the clustering state s_t^c with s_{t+1}^c.
Obtain all optimum evaluation functions?
No
Yes
END

Fig. 3 Framework of BQ_ASCP procedure

Table 1 Main parameters for Example 1

M	N	λ	u_{p1}	u_{p2}	u_{f1}	u_{f2}	ε	γ
6	2,400	1/5.5	2	13	1	6	0.15	0.7

a_v via ε-*greedy* method, which is performed by the processing agent, that is $a_t = a_v$, $a_v \in A$.

Step 7 Detect the state s_{t+1} that KMC stays at time t+1. The clustering state s^c_{t+1} is obtained by calculating dissimilarity measure. B–Q learner receives an immediate reward r_{t+1}, then the evaluation function $Q_n\left(s^c_u, a_v\right)$ is calculated through Eq. 6.

Step 8 Substitute the clustering state s^c_t with s^c_{t+1} and repeat steps 6–8 until all optimum evaluation functions $Q^*\left(s^c_u, a_v\right)$ are obtained.

Algorithm 1 can be applied to any dynamic scheduling problem. Once all optimum evaluation functions are obtained, the most effective scheduling knowledge are acquired and used to conduct the processing agents in KMC.

3.3 Control policy (BQ_ASCP)

Based on Algorithm 1, the physical framework of adaptive scheduling control policy of KMC is shown in Fig. 2. It consists of the KMC simulator, B–Q learner, scheduling decision-making module, and dispatching rule base.

From Fig. 2, the principle of BQ_ASCP (B–Q algorithm-based adaptive scheduling control policy) is as follows: The KMC simulator operation yields a series of KMC simulation states on the basis of the scheduling rules in the dispatching rule base. Then, system clustering states are obtained by the B–Q learner. When the job reaches KMC and is processed, the learner will detect the current system state and start learning by using B–Q learning algorithm to acquire the dynamic scheduling knowledge of the system, thus updating the knowledge in the scheduling knowledge base. When a certain processing agent is idle and there are unfinished jobs in its buffer, the decision-maker will read the scheduling knowledge in the scheduling knowledge base by the detected state and select suitable scheduling rules to dispatch the jobs to the processing agent, ensuring the smooth running of KMC. The framework of the BQ_ASCP procedure is schematically illustrated in Fig. 3.

Hence, BQ_ASCP acquires the new scheduling knowledge mainly by learning constantly and selects dispatching rules dynamically according to KMC states. This control policy has a strong adaptability to frequent disturbance in the dynamic scheduling. It is important to note that the B–Q learner can accomplish learning and update the scheduling knowledge base by offline learning.

4 Case study

In this section, different scheduling methods of job shop, namely, BQ_ASCP, earliest due date (EDD), shortest processing time (SPT), and minimum slack time (MST), are compared under the performance criterion, i.e., mean tardiness. Simulation experiment is designed for a KMC consisting of M processing agents, and there are N jobs to be processed. The processing agent can only deal with one job at a time. Job arrivals are generated through a negative exponential distribution, and the average arrival rate of jobs is λ. Arriving jobs have from one to six operations, and the total number k_j of operations for jth job is equally distributed among the integers from 1 to 6. The routing for each job is generated randomly, with every processing agent having an equal probability of being chosen. Similarly, the first operation of an arriving job is equally likely to require any one of the processing agents. No successive pair of operations requires the same processing agent.

In the dispatching rule base, three popular dispatching rules are chosen, i.e., EDD, SPT, and MST. In the light of the existing literature, one can find that the performance of dispatching rules EDD, SPT, and MST varies with the state features, namely, mean allowance factor, system utilization, relative machine workload, and mean slack time. Therefore,

Table 2 Mean tardiness from different approaches in Example 1

Approach	Mean tardiness of 50 episodes										Average value (500 episodes)
	1	2	3	4	5	6	7	8	9	10	
EDD	10.271	9.863	10.458	10.447	10.576	10.335	10.299	11.459	9.673	9.618	10.300
MST	10.047	10.642	11.171	9.687	9.658	9.989	10.813	10.910	11.308	10.550	10.478
SPT	16.096	15.593	17.148	15.592	14.872	16.595	14.591	16.408	16.480	16.268	15.964
BQ_ASCP	9.628	8.785	7.925	9.527	9.405	9.429	9.134	8.858	9.018	9.060	9.045

the four state features are employed to describe the KMC state in the paper. In the experiment, we set the due dates d_j for the jth job as follows:

$$d_j = rt_j + f_j \sum_{q=1}^{k_j} p_{jq} \qquad (12)$$

where p_{jq} denotes the time required for operation q on job j and rt_j is the time when the job reaches KMC. Allowance factor f_j is subject to uniform distribution, i.e., $f_j \sim \mathrm{U}(u_{f1}, u_{f2})$.

The objective function is aimed at minimizing the mean tardiness of jobs. However, the B–Q learning algorithm converges at the maximum value. Thus, it is multiplied by a minus for the maximum problem conversion, and the immediate reward r is set as follows:

$$r = \begin{cases} -(C_j - d_j) & \text{tardines for job} j \\ 1 & \text{no tardines for job} j \end{cases} . \qquad (13)$$

where C_j is the actual completion time for the job j.

Example 1. It is assumed that each job processing time follows uniform distribution $\mathrm{U}(u_{p1}, u_{p2})$. The main parameters for the experiment are listed in Table 1.

When KMC finishes processing 2,400 jobs, an episode is completed. The simulation experiment is executed by Matlab 6.5 language and is run on a Pentium IV 2.4 GHz personal computer running Windows 2000. The results of 500 episodes are obtained following the rules like EDD, SPT, MST rules, and BQ_ASCP. Considering the impact of many stochastic factors accordingly, mean tardiness (MT) of jobs in 50 episodes are compared with one another, as shown in Table 2, where the *average value* is the mean tardiness of jobs in 500 episodes. The variation of mean

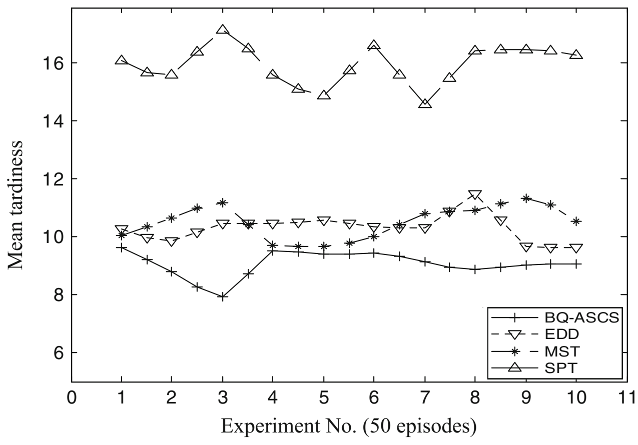

Fig. 4 Variation of mean tardiness from different approaches in Example 1

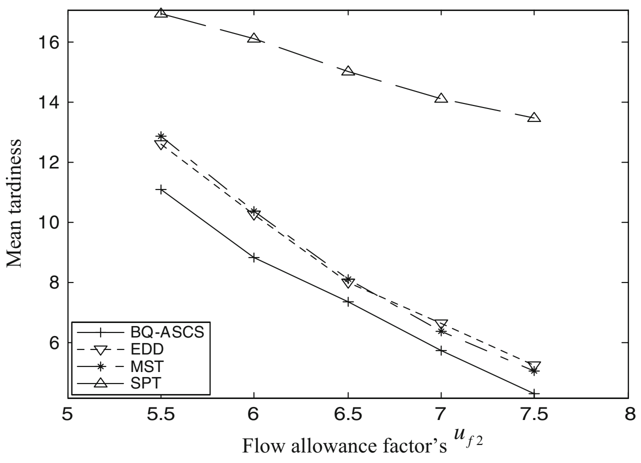

Fig. 5 Mean tardiness from different approaches with variant allowance factor

tardiness per 50 episodes from different approaches is presented in Fig. 4.

To illustrate the due date impact on mean tardiness from BQ_ASCP, sets u_{f1}=1 and u_{f2}=5.5, 6, …, 7.5, respectively, the simulation experiments with 200 episodes are implemented. Considering the influence of stochastic factor as well, we make comparisons among mean tardiness from different approaches, and the results are shown in Fig. 5.

From Fig. 4, it can be seen that BQ_ASCP method performs better on the mean tardiness than others. From the mean tardiness of 500 episodes in Table 2, it can be concluded that the mean tardiness for BQ_ASCP is 12.18% lower than that for the best EDD rule and 43.34% improvement over the worst SPT rule. Likewise, Fig. 5 shows that the BQ_ASCP approach outperforms the other three dispatching rules on mean tardiness performance when allowance factors with different urgency are taken into account, and it can reduce the mean tardiness significantly for a different due date of a job.

Example 2. To verify the validity of the proposed BQ_ASCP approach in different manufacturing environments, each job processing time is supposed to be subject to an exponential distribution with a mean of λ_1. The main experiment parameters are shown in Table 3, and a total of 500 episodes are executed. Similar to Example 1, considering

Table 3 Main parameters for Example 2

M	N	λ	λ_1	u_{f1}	u_{f2}	ε	γ
6	2,400	1/5.5	6.5	1	6	0.15	0.7

Table 4 Mean tardiness from different approaches in Example 2

Approach	Mean tardiness of 50 episodes										Average value (500 episodes)
	1	2	3	4	5	6	7	8	9	10	
EDD	6.340	6.647	6.505	7.209	7.141	6.541	6.789	7.137	6.794	7.074	6.818
MST	7.075	7.215	7.387	7.563	6.908	7.264	7.574	6.977	7.402	7.869	7.323
SPT	6.425	6.455	6.330	6.398	6.439	6.532	6.286	6.309	6.045	6.471	6.369
BQ_ASCP	5.759	5.571	5.784	5.720	5.398	5.610	5.882	5.797	5.679	5.725	5.693

the influence of stochastic factors during processing, a mean tardiness (MT) per 50 episodes is compared respectively between the BQ_ASCP method and the other three dispatching rules as shown in Table 4 and Fig. 6.

From the average values in Table 4, we can conclude that the BQ_ASCP approach improves the mean tardiness by 10.61% and 22.26%, respectively, in comparison with the best SPT or worst MST rules constantly. As shown in Fig. 6, the proposed BQ_ASCP performs much better than the other three rules when job processing time follows an exponential distribution.

5 Conclusions

Methods reported in literature acquire effective scheduling knowledge in an unstable and time-varying environment on the basis of precise training examples for selecting a proper dispatch rule to schedule jobs. However, in real production, it is difficult to obtain this type of precise field-expert knowledge. Focusing on the difficulty of scheduling knowledge acquisition in dynamic scheduling, this paper brings forward an adaptive B–Q learning algorithm that has the characteristic of adaptation to dynamic environment. The proposed algorithm needs no prior knowledge other than the interaction with KMC to acquire scheduling knowledge. Based on B–Q learning algorithm, BQ_ASCP is derived for selecting dispatching rules and arranges the processing sequences of the jobs in KMC according to the operating state for KMC. It can overcome frequent disturbance in dynamic scheduling system and accomplish learning and updating the scheduling knowledge by offline learning.

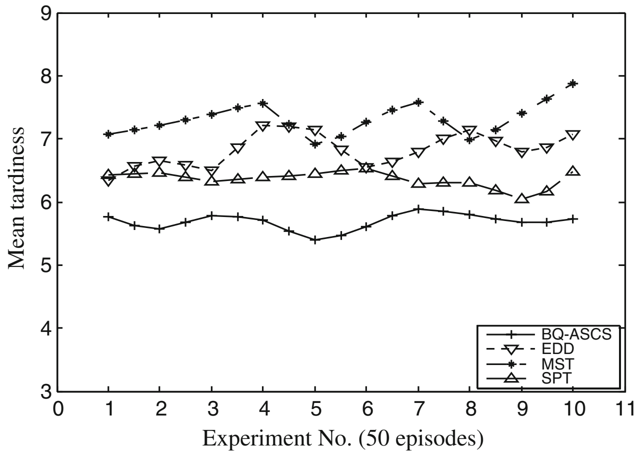

Fig. 6 Variation of mean tardiness from different approaches in Example 2

KMC is a typical discrete event dynamic system, which can comprise several hundreds or even thousands of states. As is known, traditional reinforcement learning is of structural credit assignment; i.e., when facing an environment system with large state space, it is difficult to fully explore state space for reinforcement learning. To cope with the structural credit assignment influence, basic sequential algorithmic scheme (BSAS) is employed to obtain the cluster state of KMC for successful reduction of KMC state space complexity. As a consequence of adoption of the cluster state, the value of evaluation function $Q\left(s_t^c, a_t\right)$ may change drastically, which can lead to the deterioration of learning efficiency in B–Q learning algorithm, so evaluation function threshold Θ is introduced to minimize the value fluctuations of evaluation function in learning process.

The performance of several scheduling methods has been experimentally evaluated and compared. The results show that for cases of allowance factors with different urgency, the given scheduling control policy presented herein is more feasible and effective than the single dispatching rule. To simplify and clarify the problem, we selected only four state features to describe the KMC state in the paper. In fact, an accurate system state is composed of many state features, their valid selection being key to the description of the former. That is what needs further exploring in the near future.

Acknowledgment The work is supported by the National Natural Science Foundation of China under grants 60,574,062 and 50,475,075, and by the National High Tech R&D Program of China under grant 2007AA04Z112, China. We thank the editor and Professor Andrew Yeh-Ching Nee, the anonymous referees, and Professor Li Lu for valuable comments and suggestions.

References

1. Devedzic V, Radovic D (1999) A framework for building intelligent manufacturing system. IEEE Trans Syst Man Cybern C 29(3):422–439
2. Holweg M (2007) The genealogy of lean production. J Oper Manag 25(2):38–42 doi:10.1016/j.jom.2006.04.001
3. Domico M, Saidler JM (2002) Clean, green manufacturing. Ceram Ind 152(2):54–56
4. Gunasekaran A (1999) Agile manufacturing: a framework for research and development. Int J Prod Econ 62(1/2):87–105 doi:10.1016/S0925-5273(98)00222-9
5. Jiang ZB, Fung RYK (2003) An adaptive agile manufacturing control infrastructure based on TOPNs-CS modelling. Int J Adv Manuf Technol 22(3/4):191–215 doi:10.1007/s00170-002-1459-z
6. Koufteros X, Vonderembse M, Doll W (2001) Concurrent engineering and its consequences. J Oper Manag 19(1):97–115 doi:10.1016/S0272-6963(00)00048-6
7. Qiang L, Zhang YF, Nee AYC (2001) A distributive and collaborative concurrent product design system through the WWW/Internet. Int J Adv Manuf Technol 17(5):315–322 doi:10.1007/s001700170165
8. Yan HS, Liu F (2001) Knowledgeable manufacturing system—a new kind of advanced manufacturing system. Comput Integr Manuf Syst 7(8):7–11 (in Chinese)
9. Yan HS (2006) A new complicated-knowledge representation approach based on knowledge meshes. IEEE Trans Knowl Data Eng 18(1):47–62 doi:10.1109/TKDE.2006.2
10. Yan HS, Dong H (2007) A new approach to modeling, control and simulation of knowledgeable manufacturing cell. Int J Prod Res 45(17):3779–3808 doi:10.1080/00207540500497082
11. Xue CG, Yan HS (2004) A study on self-reconfiguration of a knowledgeable manufacturing system. Proc Instit Mech Eng B J Eng Manuf 218(11):1601–1617
12. Zhang WJ, Freiheit T, Yang HS (2005) Dynamic scheduling in flexible assembly system based on timed Petri nets model. Robot Comput Integr Manuf 21(6):550–558 doi:10.1016/j.rcim.2004.12.002
13. Shnits B, Rubinovitz J, Sinreich D (2004) Multicriteria dynamic scheduling methodology for controlling a flexible manufacturing system. Int J Prod Res 42(17):3457–3472 doi:10.1080/00207540410001699444
14. Choi BK, You NK (2006) Dispatching rules for dynamic scheduling of one-of-a-kind production. Int J Comput Integr Manuf 19(4):383–392 doi:10.1080/09511920500407541
15. Diaz AR, Tchernykh A, Ecker KH (2003) Algorithms for dynamic scheduling of unit execution time tasks. Eur J Oper Res 146 (2):403–416 doi:10.1016/S0377-2217(02)00236-9
16. Cowling PI, Ouelhadj D, Petrovic S (2004) Dynamic scheduling of steel casting and milling using multi-agents. Prod Plan Control 15(2):178–188 doi:10.1080/09537280410001662466
17. Zhou ZD, Wang HH, Chen YP, Ong SK, Fuh JYH, Nee AYC (2003) A multi-agent-based agile scheduling model for a virtual manufacturing environment. Int J Adv Manuf Technol 21 (12):980–984 doi:10.1007/s00170-002-1420-1
18. Priore P, Fuente D, Pino R, Puente J (2003) Dynamic scheduling of flexible manufacturing systems using neural networks and inductive learning. Integr Manuf Syst 14(2):160–168 doi:10.1108/09576060310459456
19. Branke J, Mattfeld DC (2005) Anticipation and flexibility in dynamic scheduling. Int J Prod Res 43(15):3103–3129 doi:10.1080/00207540500077140
20. Chou FD, Chang PC, Wang HM (2006) A hybrid genetic algorithm to minimize makespan for the single batch machine dynamic scheduling problem. Int J Adv Manuf Technol 31(3/4):350–359 doi:10.1007/s00170-005-0194-7
21. Baker KR (1984) Sequencing rules and due-date assignments in a job shop. Manage Sci 30(9):1093–1104
22. Arzi Y, Iaroslavitz L (1999) Neural network-based adaptive production control system for a flexible manufacturing cell under a random environment. IIE Trans 31(3):217–230
23. Wan GH (1999) Fuzzy logic system for dynamic job shop scheduling. Proc IEEE Int Conf Syst Man Cybern 4:546–551
24. Piramuthu S, Shaw M, Fulkerson B (2000) Information-based dynamic manufacturing system scheduling. Int J Flex Manuf Syst 12(2/3):219–234 doi:10.1023/A:1008151831821
25. Koulamas C (1994) The total tardiness problem: review and extensions. Oper Res 42(6):1025–1041
26. Kim CO, Jun J, Baek JK, Smith RL, Kim YD (2005) Adaptive inventory control models for supply chain management. Int J Adv Manuf Technol 26(9/10):1184–1192 doi:10.1007/s00170-004-2069-8
27. McDonnell P, Joshi S, Qiu RG (2005) A learning approach to enhancing machine reconfiguration decision-making games in a heterarchical manufacturing environment. Int J Prod Res 43 (20):4321–4334 doi:10.1080/00207540500142431
28. Theodoridis S, Koutroumbas K (2003) Pattern recognition, 2nd edn. Academic, San Diego, CA, USA