文章编号: 1001-0920(2007)06-0675-05

强化学习和仿真相结合的车间作业排序系统

潘燕春, 冯允成, 周　泓, 魏佳呈
(北京航空航天大学 经济管理学院, 北京 100083)

摘　要: 设计了一个强化学习和仿真相结合的动态实时车间作业排序系统. 首先引入多个随机变量,将车间作业排序问题转换成序贯决策问题;然后通过仿真手段构建车间作业排序问题的模型环境,求取系统性能指标并保证解的可行性;接着设计了一个多智能体 Q-学习算法和仿真集成解决作业排序问题;最后通过仿真优化实验验证了该系统的有效性.

关键词: 强化学习; 仿真; 序贯决策; 车间作业排序
中图分类号: TP391.9　　**文献标识码**: A

Reinforcement learning integrated with simulation for job-shop scheduling system

PAN Yan-chun, FENG Yun-cheng, ZHOU Hong, WEI Jia-cheng
(School of Economics and Management, Beihang University, Beijing 100083, China. Correspondent: PAN Yan-chun, E-mail: pan_yc@163.com)

Abstract: A dynamic and real-time system integrating reinforcement learning with simulation is designed for job-shop scheduling. Several stochastic variables are introduced to transform the job-shop scheduling problem into sequential decision problem. The model environment of job-shop scheduling is built by simulation for obtaining the system performance indices and ensuring the feasibility of the solution. Then, a multi-agent Q-learning algorithm integrated with simulation is developed to solve the job-shop problem. Finally, simulation and optimization experiments show the effectiveness of the system.

Key words: Reinforcement learning; Simulation; Sequential decision; Job-shop scheduling

1 引　　言

车间作业排序问题是生产系统和服务系统中的一类典型问题,也是众多学者研究的一个传统和热点问题. 一般描述如下:有 m 个资源, n 个作业,各作业在资源上的处理路径和时间(或成本)可能不同,研究目标是确定每个资源上的作业顺序和开工时间,使得有关评价指标达到最优,如最大完工时间、总费用、总延迟等[1]. 这类问题已证明是 NP 难题,具有相当大的计算复杂性,随着问题规模的增加,计算难度呈指数增长,传统的解析方法已经难于甚至根本不可能对其进行求解[2].

处理这类问题的常用方法就是现代启发式优化算法,如遗传算法、禁忌搜索、神经网络等. 但是这些方法一般都要求事先知道车间作业排序问题的所有信息,即对作业进行静态排序. 然而现实系统却存在很多随机因素,如作业到达时间的不确定性、处理时间的不确定性等,因此动态的实时的排序系统将更加符合实际[3].

近年来,人工智能和机器学习领域的一个既崭新又古老的课题 ——强化学习[4,5]得到了研究人员的广泛重视,但其应用目前还主要集中在游戏比赛、控制系统和机器人领域[6-8],在作业排序等管理问题上的应用尚不多见. Wang 等[9,10]在单机作业排序问题上使用 Q-学习算法,发现智能体能够从给定的分派规则中选择出较好的分派规则,由此说明了强化学习应用于作业排序问题的可能性和有效性.

本文从序贯决策的角度出发,考虑离散组合优化的车间作业排序问题,通过仿真的手段构建车间

收稿日期: 2006-03-01; **修回日期**: 2006-06-01.
基金项目: 国家自然科学基金项目(70371005,70521001); 高等学校博士学科点专项科研基金项目(20020006-4); 新世纪优秀人才支持计划(NCET-04-0175).
作者简介: 潘燕春(1979—),男,江西赣州人,讲师,博士,从事生产管理、序贯决策等研究; 冯允成(1933—),男,上海人,教授,博士生导师,从事系统仿真、优化理论和虚拟现实等研究.

作业排序系统环境,采用 Q-学习算法和仿真环境交互,最终设计出了一个多智能体和仿真相结合的动态车间作业排序系统(MAQLS).仿真优化实验结果验证了该系统的有效性.

2 车间作业排序问题的序贯决策求解

2.1 序贯决策和强化学习

序贯决策又称动态决策、多阶段决策、马尔可夫决策,与普通决策相比,其决策过程可以分为若干个相互联系的阶段,在它的每一个阶段都需要做出决策.各阶段的决策既依赖于当前面临的状态,又影响以后的发展.决策的结果最终将形成一个序列,决策目标就是要确定各个状态的决策,从而使整个序列达到最优[11].

强化学习问题[4,5]本身就是动态的、实时的、多阶段的决策问题.强化学习智能体的目标,就是要学习一个控制策略,以最大化所有状态(或"状态-行动"对)的期望累积报酬.与传统的解决序贯决策问题的动态规划和马尔可夫决策过程等方法相比,强化学习算法有相对标准的模型和结构,只要环境构造合理,就可以容易地避免转移概率和收益矩阵的计算,从而避免"维数灾难"问题.

2.2 车间作业排序问题的序贯决策求解

针对车间作业排序问题,从序贯决策的角度考虑,采用强化学习方法,可以将每一个资源看成一个自主的智能体.为了减小智能体的学习空间,提高收敛速度,同时又不影响对问题的求解,引入如下随机变量:$x_t(i)$ 表示在 t 时刻有无作业到达机器 i,有则取值为1,无则取值为0;$y_t(i)$ 表示在 t 时刻机器 i 的状态,繁忙时取值为1,空闲时取值为0;$w_t(i)$ 表示在 t 时刻机器 i 上的作业队列长度.

假定 $u_t(i) = x_t(i) + y_t(i)$,定义时刻 t 智能体 i 所处的环境状态为 $s_t(i) = (u_t(i), w_t(i))$,容易得到如下结论:1) 当 $w_t(i) > 0$ 时,机器 i 必然处于繁忙状态,即 $y_t(i) = 1$,则有 $u_t(i) > 0$;2) 当 $u_t(i) = 0$ 时,必然存在 $w_t(i) = 0$.

长期以来,学者们对分派规则作了大量研究,发现用这些分派规则来产生作业排序能够取得很好的效果,而且针对不同的评价指标,可以采取不同的分派规则以达到更加理想的结果[12-14].因此,可以以事先设定好的分派规则作为智能体的行动.针对车间作业排序问题的上述变换实际上是采用了强化学习的状态聚合和函数近似技术[4],只要保证分派规则的多样性使智能体在采取行动时选择任意一个作业都是有可能的,则不会影响算法的收敛性.

当环境状态发生变化时,智能体将根据当前状态进行决策,即确定采取何种分派规则来选择作业进行处理,这又分为以下两种情况(以任意一个智能体 i 在时刻 t 为例):

1) 有新的作业 J_t 到达,则 $x_t(i) = 1$, $u_t(i) \geq 1$.如果此时 $u_t(i) + w_t(i) < 2$,则有 $w_t(i) = y_t(i) = 0$, $u_t(i) = x_t(i) = 1$,智能体 i 直接处理 J_t;如果此时 $u_t(i) + w_t(i) \geq 2$,结合结论1),必有 $y_t(i) = 1$,即机器繁忙,设此时正在处理的作业为 J_t',则智能体 i 先根据当前状态确定分派规则,再由分派规则确定应该选择 J_t 和 J_t' 中的哪一个作业进行处理.

2) 资源完成某项作业.如果此时 $w_t(i) = 0$,则智能体 i 不作决策;如果此时 $w_t(i) = 1$,则智能体 i 将选择作业队列中的唯一作业进行处理;如果此时 $w_t(i) > 1$,则智能体 i 将先根据当前状态确定分派规则,再由分派规则确定应该选择作业队列中的哪个作业进行处理.

这样构建的排序系统就可以根据作业的随机到达和离开动态实时地确定下一个要处理的作业,并且可以把车间作业排序问题转换成序贯决策问题,使用强化学习方法进行求解.

3 MAQLS系统设计

3.1 整体设计

MAQLS的整体框架如图1所示,其中"状态-行动"对信息包括资源标识、状态、行动、Q值、"状态-行动"对被选取的次数、本次循环中是否采取了该"状态-行动"对等信息.自主智能体根据仿真模型环境给定的当前状态 s,结合有关"状态-行动"对信息,通过策略 π 产生行动 a,执行该行动后仿真模型环境将报酬值 r 反馈给自主智能体,由策略 π 根据该报酬值更新有关"状态-行动"对信息.一个学习周期结束即可以得到问题的一个解,仿真模型环境再将解的有关信息反馈给全局智能体,由策略 π_g 更新有关"状态-行动"对信息,从而指导自主智能体的学习过程.

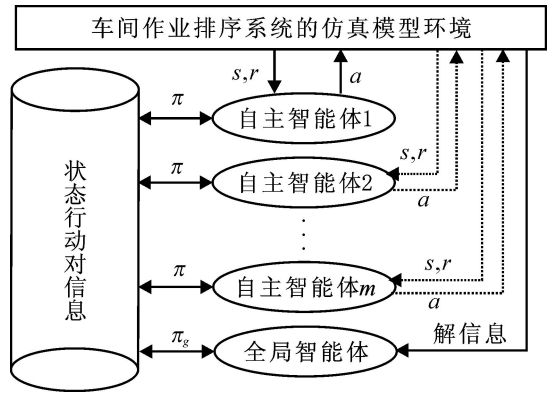

图1 MAQLS整体框架

3.2 仿真模型设计

本文通过仿真求取系统的性能指标并验证和

确保解的可行性. 利用虚拟抢占规则[15,16] 不仅实现了对作业排序的仿真建模,还能够将仿真产生的非延迟排序转换成活动排序,从而不会遗漏最优解[1].

3.3　多智能体 Q- 学习算法设计

Q- 学习算法[4,5] 是强化学习算法中的一类典型算法,它用"状态 - 行动"对的 Q 值对累积报酬进行估计,

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]. \quad (1)$$

其中:$\alpha \in [0,1]$ 为学习率;$\gamma \in [0,1]$ 为折扣率;s' 为智能体在状态 s 下采取行动 a 系统到达的新状态;r 为智能体在状态 s 下采取行动 a 到达新状态 s' 时获得的立即报酬,由报酬函数确定.

ε 贪婪行动值法[4,5] 是智能体选择行动的一种重要方法,其思想在于:智能体以一个较大的概率 $(1-\varepsilon)$ 选取完全贪婪行动 $a^* = \arg\max_a Q(s,a)$,以一个较小的概率(贪婪率) ε 随机选取行动. 一般推荐贪婪率取常值 0.1.

设定所有的自主智能体都是等价的,具有相同的行动集合和策略,但具有独立的"状态 - 行动"对信息. 自主智能体采取 Q- 学习算法对"状态 - 行动"对空间进行优化搜索,采取"ε 贪婪行动值法"选取行动. 为了避免算法陷入局部最优,提出了一种贪婪率周期性递减的策略,如图 2 所示.

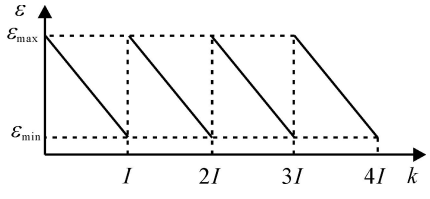

图 2　贪婪率周期性递减策略

$$\varepsilon = \varepsilon_{\max} - \frac{(\varepsilon_{\max} - \varepsilon_{\min})(k \% I)}{I}. \quad (2)$$

其中:贪婪率 $\varepsilon \in [0,1]$ 越小算法越贪婪,越能充分利用现有的学习成果,ε 越大算法越随机,越有利于探索新的有效策略;$\varepsilon_{\max}$ 和 $\varepsilon_{\min}$ 分别为最大贪婪率和最小贪婪率;I 为迭代代数基准;k 为当前迭代代数;$(k \% I)$ 表示 k 除以 I 的余数. 算法先迭代 I 代,如果有改进再新增 I 代循环,如果在新增的 I 代循环中算法都没有改进则终止. 在每个 I 代循环中,贪婪率都从 $\varepsilon_{\max}$ 线性递减到 $\varepsilon_{\min}$. 这样设计的贪婪率具有一定的自适应性,更有可能跳出局部最优,从而在深度搜索和广度搜索中找到一个比较好的均衡.

4　仿真实验

为了说明所提算法的有效性,本文采用 OR-Library[17] 收录的 82 个车间作业排序问题的国际标准算例进行验证. 这些算例目前只提供了系统最大完工时间 $C_{\max}$ 作为评价标准,而且公布的最优结果可能是不同学者采用不同算法获得的.

4.1　行动集合

以分派规则作为智能体的行动. 如果分派规则过多,将使"状态 - 行动"对空间剧增,导致算法难以收敛;如果分派规则过少,将使智能体在采取行动时选择任意一个作业的可能性减小,导致算法遗漏最优解. 针对系统最大完工时间评价指标,根据有关经验信息[12-14],本文设计了如表 1 所示的 9 个不同的分派规则,并且给它们赋予了相应的初始 Q 值 (Q_0). Q 值越大对应的分派规则被选取的概率越大.

表 1　行动集合

行动	描　　述	Q_0
LPT	最长加工时间优先	1
FIFO	先到先加工	2
LNMRT	下一道操作机器剩余的总加工时间最长优先	3
SNMRT	下一台机器剩余的总加工时间最小优先	4
SNMU	下一台机器已经完成的总加工时间最小优先	5
SRT	最短剩余加工时间优先	6
SWINQ	下一台机器的队列的总加工时间最小优先	7
LRT	最长剩余加工时间优先	8
SPT	最短加工时间优先	9

4.2　报酬函数

在 Q- 学习中,要使最大化 Q 值和最小化系统最大完工时间的方向保持一致,智能体报酬函数的设计至关重要. 必须尽量保证使系统最大完工时间越小的"状态 - 行动"对的报酬值越大.

假设在一个学习周期某时点 t 的所有机器部分序的最大完工时间为 $C^1_{\max}(t)$,此时某智能体 i 在状态 $s_t(i)$ 下采取了某个行动 $a_t(i)$. 由该行动确定下一个要处理的作业 J_t,当 J_t 加工完成时,得到其处理时间 p_t,并得到一个新的所有机器部分序,其最大完工时间为 $C^2_{\max}(t)$. 则设定该"状态 - 行动"对的报酬为

$$r_t(i) = p_t - (C^2_{\max}(t) - C^1_{\max}(t)), \quad (3)$$

即优先处理使系统最大完工时间增量较小的零件和加工时间较大的零件. 这样设计的报酬函数有利于最小化系统最大完工时间,使智能体朝着最优化目标调整行为策略.

4.3　全局智能体行动策略

假设智能体在当前学习周期得到的系统最大完工时间为 $C_{\max}$,前一个学习周期得到的系统最大完工时间为 $C'_{\max}$,设定指导信息为 $\Delta = C'_{\max} - C_{\max}$. 全局智能体的行动策略为:当"状态 - 行动"对 (s,a) 在当前学习周期中被采取时,则用

$$Q(s,a) \leftarrow Q(s,a) + \Delta \quad (4)$$

更新其 Q 值;如果 $C_{\max} < C'_{\max}$,则 $\Delta > 0$,此时"状态

- 行动”对的 Q 值将得到增加;如果 $C_{\max} > C'_{\max}$,则 $\Delta < 0$,此时“状态 - 行动”对的 Q 值将减小. 由此便可从全局来指导自主智能体朝着 $C_{\max}$ 下降的方向采取行动.

4.4 仿真优化实验

根据经验信息[4] 设定算法的折扣率为 $\gamma = 0.9$, 学习率为

$$\alpha = \frac{1}{1 + \beta N(s,a)}. \tag{5}$$

其中: $N(s,a)$ 表示“状态 - 行动”对 (s,a) 出现的次数;$\beta > 0$ 为学习率系数,β越大学习率下降越快,算法收敛速度越快,但越有可能收敛到局部最优解. 设

$$\Delta^q_{\max} = \max_{(s,a)} |\Delta Q(s,a)|, \tag{6}$$

表示当前学习周期中所有“状态 - 行动”对的最大 Q 差值,设定阈值为 $\theta(\theta > 0)$,当 $\Delta^q_{\max} < \theta$时,认为算法收敛. θ越小算法越精确,但收敛速度越慢.

以 la01 为参数实验算例,最大贪婪率 $\varepsilon_{\max}$ 的取值集合设定为 $\{0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9\}$,设定最小贪婪率为一个足够小的值 $\varepsilon_{\min} = 0.05$. 学习率系数 β的取值集合设定为$\{1,2,3,4,5\}$,阈值 θ的取值集合设定为$\{0.5,1\}$. 对某参数进行实验时,固定该参数值,任意调整其余两个参数的组合. 经过初步实验,确定迭代代数基准 $I = 5\,000$. 参数实验的统计结果如表 2 所示,实验结果和理论分析非常吻合. 综合效果和效率两个方面的因素,在以下的仿真优化实验中,参数取值为:$\varepsilon_{\max} = 0.6$,β

表 2 参数实验结果统计

参数值	达优比例	平均代数	参数值	达优比例	平均代数
$\varepsilon_{\max} = 0.1$	1/10	2 194	$\varepsilon_{\max} = 0.9$	9/10	4 021
$\varepsilon_{\max} = 0.2$	3/10	2 597	$\beta = 1$	16/18	3 696
$\varepsilon_{\max} = 0.3$	6/10	1 793	$\beta = 2$	10/18	3 307
$\varepsilon_{\max} = 0.4$	5/10	3 128	$\beta = 3$	12/18	3 002
$\varepsilon_{\max} = 0.5$	5/10	3 398	$\beta = 4$	7/18	2 715
$\varepsilon_{\max} = 0.6$	7/10	2 771	$\beta = 5$	3/18	2 361
$\varepsilon_{\max} = 0.7$	6/10	3 113	$\theta = 0.5$	30/45	3 352
$\varepsilon_{\max} = 0.8$	6/10	4 269	$\theta = 1$	18/45	2 988

注:达优比例中的分母为所有的参数组合数,分子为获得了最优解的参数组合数.

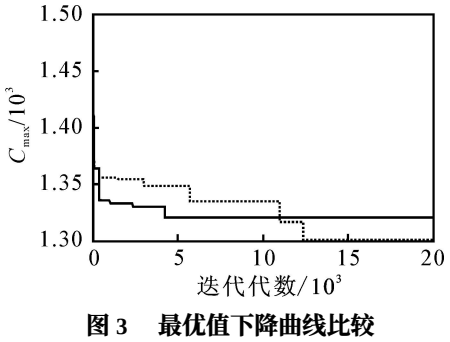

图 3 最优值下降曲线比较

表 3 算例规模

规 模	算 例	规 模	算 例
6 ×6	ft06	20 ×10	la26-30 ,swv01-05
10 ×5	la01-05	30 ×10	la31-35
15 ×5	la06-10	50 ×10	swv11-20
20 ×5	ft20 ,la11-15	15 ×15	la36-40
10 ×10	abz5 ,abz6 ,ft10 , la16-20 ,orb01-10	20 ×15	abz7 ,abz8 ,abz9 , swv06-10
15 ×10	la21-25	20 ×20	yn1-4

注:规模中的第 1 个数值为工件数,第 2 个数值为机器数.

表 4 算例的仿真优化结果

算例	C^{**}	C^*	误差	算例	C^{**}	C^*	误差
abz5	1 234	1 270	2.917	la34	1 721	1 765	2.557
abz6	943	980	3.924	la35	1 888	1 888	0
abz7	656	679	3.506	la36	1 268	1 368	7.886
abz8	669 *	750	12.108	la37	1 397	1 507	7.874
abz9	679 *	774	13.991	la38	1 196	1 315	9.950
ft06	55	55	0	la39	1 233	1 285	4.217
ft10	930	1 021	9.785	la40	1 222	1 320	8.020
ft20	1 165	958	6.208	orb01	1 059	1 114	5.194
la01	666	666	0	orb02	888	954	7.432
la02	655	669	2.14	orb03	1 005	1 048	4.279
la03	597	619	3.685	orb04	1 005	1 037	3.184
la04	590	615	4.237	orb05	887	941	6.088
la05	593	593	0	orb06	1 010	1 064	5.347
la06	926	926	0	orb07	397	423	6.549
la07	890	890	0	orb08	899	980	9.010
la08	863	863	0	orb09	934	975	4.390
la09	951	951	0	orb10	944	1 038	9.958
la10	958	958	0	swv01	1 418 *	1 614	13.822
la11	1 222	1 222	0	swv02	1 475	1 603	8.678
la12	1 039	1 039	0	swv03	1 398 *	1 630	16.595
la13	1 150	1 150	0	swv04	1 483 *	1 700	14.633
la14	1 292	1 292	0	swv05	1 434 *	1 621	13.040
la15	1 207	1 243	2.983	swv06	1 696 *	1 983	16.922
la16	945	964	2.011	swv07	1 620 *	1 908	17.778
la17	784	821	4.719	swv08	1 770 *	2 083	17.684
la18	848	894	5.425	swv09	1 663 *	1 956	17.619
la19	842	875	3.919	swv10	1 773 *	2 014	13.593
la20	902	958	6.208	swv11	3 005 *	3 592	19.534
la21	1 046	1 145	9.465	swv12	3 038 *	3 613	18.927
la22	927	1 018	9.817	swv13	3 146 *	3 684	17.101
la23	1 032	1 032	0	swv14	2 968	3 220	8.491
la24	935	1 019	8.984	swv15	2 940 *	3 351	14.000
la25	977	1 073	9.826	swv16	2 924	2 924	0
la26	1 218	1 281	5.172	swv17	2 794	2 794	0
la27	1 235	1 352	9.474	swv18	2 852	2 852	0
la28	1 216	1 301	6.99	swv19	2 843	2 843	0
la29	1 153 *	1 303	13.010	swv20	2 823	2 823	0
la30	1 355	1 447	6.790	yn1	888 *	969	9.122
la31	1 784	1 784	0	yn2	909 *	999	9.901
la32	1 850	1 850	0	yn3	894 *	980	9.620
la33	1 719	1 719	0	yn4	972 *	1 083	11.420

注:C^{**} 为目前最优值,C^* 为 MAQLS 算法取得的最优值,误差为 $100 \times (C^* - C^{**})/C^{**}$.

= 1,θ = 0.5. 以中等规模算例 la28 为例,图 3 所示为固定贪婪率策略和贪婪率周期性递减策略的最优值下降曲线比较图. 图中实线为贪婪率固定,ε = 0.1,θ = 0.5,β = 1;虚线为贪婪率周期性递减,ε_{max} = 0.6,I = 5 000,β = 1,ε_{min} = 0.05,θ = 0.5. 从图 3 中可以看出,虽然固定贪婪率为推荐值 0.1 时,开始阶段最优值下降较快,但是容易陷入局部最优;而本文提出的贪婪率周期性递减策略在开始阶段最优值下降较慢,但很有可能跳出局部最优,从而寻找到更好的解.

表 3 为算例规模表. 各算例的最终仿真优化结果如表 4 所示,统计结果如表 5 所示.

表 5　算例的仿真优化结果统计

误差	个数	比例	误差	个数	比例
0	22	26.829	(10,15]	9	10.976
(0,5]	15	18.293	(15,20]	8	9.756
(5,10]	28	34.146			

从表 4 和表 5 可以看出,MAQLS 算法除了对求解难度较大的 swv01-swv15 等算例表现一般以外,对其余算例求解结果都比较理想,即使对其他求解难度较大的算例,相对误差一般也能控制在 5 % ～ 10 % 之间. 因为本文提出的 MAQLS 算法并没有针对具体的单个算例利用其特定信息进行优化,所以该算法具有普遍适用性,而且满足动态的实时的作业排序要求. 仿真优化结果充分验证了算法的有效性.

5　结　　语

强化学习算法是一种重要的机器学习方法,在游戏比赛、控制系统和机器人领域都得到了广泛的应用,但在解决诸如车间作业排序等管理难题上的有效性并没有充分发掘出来. 本文采用多智能体 Q-学习算法和仿真相结合来求解车间作业排序问题,实验结果表明了其可行性和有效性,同时也为进一步研究如何提高强化学习的效果和效率以解决更加复杂的动态随机生产计划问题提供了一个很好的基础. 如果能够设计更加有效的报酬函数、分派规则和行动策略,该算法将有很大的潜力,这也是一个很值得深入研究的内容.

参考文献(References)

[1] French S. Sequencing and scheduling: An introduction to the mathematics of job shop[M]. West Sussex: Ellis Horwood Limited Company, 1982.

[2] Garey M R, Johnson D S, Sethi R. The complexity of flowshop and jobshop scheduling [J]. Mathematics of Operations Research, 1976, 1(2): 117-129.

[3] Aydin M Emin, Oztemel Ercan. Dynamic job-shop scheduling using reinforcement learning agents [J]. Robotics and Autonomous Systems, 2000, 33(2/3): 169-178.

[4] Richard S Sutton, Andrew G Barto. Reinforcement learning: An introduction [M]. Cambridge: MIT Press, 1998.

[5] Tom M Mitchell. Machine learning[M]. New York: McGraw-Hill Press, 1997.

[6] Tesauro G J. Temporal difference learning and TD-gammon[J]. Communications of the ACM, 1995, 38(3): 58-68.

[7] Sebastian T, Mitchell T M. Lifelong robot learning[J]. Robotics and Autonomous System, 1995, 15(1/2): 25-46.

[8] Berebji H R. Learning and tuning fuzzy logic controllers through reinforcements [J]. IEEE Trans on Neural Networks, 1992, 3(5): 724-740.

[9] Wang Y C, Usher John M. Learning policies for single machine job dispatching [J]. Robotics and Computer-integrated Manufacturing, 2004, 20(6): 553-562.

[10] Wang Y C, Usher John M. Application of reinforcement learning for agent-based production scheduling[J]. Engineering Applications of Artificial Intelligence, 2005, 18(1): 73-82.

[11] Littman Michael Lederman. Algorithms for sequential decision-making [J]. Dissertation Abstracts International, 1996, 57(10): 63-69.

[12] Holthaus Oliver, Rajendran Chandrasekharan. Efficient dispatching rules for scheduling in a job shop [J]. Int J of Production Economics, 1997, 48(1): 87-105.

[13] Holthaus O. Design of efficient job shop scheduling rules[J]. Computers and Industrial Engineering, 1997, 33(1/2): 249-252.

[14] Mohanasundaram K M, Natarajan K, Viswanathkumar G, et al. Scheduling rules for dynamic shops that manufacture multi-level jobs [J]. Computers and Industrial Engineering, 2003, 44(1): 119-131.

[15] Pan Y C, Feng Y C. A new heuristic algorithm for job shop scheduling [C]. Proc of the 7th Int Conf on Industrial Management. Beijing: China Aviation Industry Publisher, 2004: 191-196.

[16] Pan Y C, Zhou H, Feng Y C, et al. Modeling and simulation optimization systems design for job shop problems based on arena[C]. System Simulation and Scientific Computing. Beijing: International Academic Publishers, 2005: 1393-1397.

[17] Dirk C M, Rob J M, The international standard testing cases on scheduling problems from OR-library [DB/OL]. http://www.ms.ic.ac.uk/info.htm, 1996.