Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

A REINFORCEMENT LEARNING ALGORITHM TO MINIMIZE THE MEAN TARDINESS OF A SINGLE MACHINE WITH CONTROLLED CAPACITY

Hadeel D. Idrees
Mahdy O. Sinnokrot
Sameh Al-Shihabi

Industrial Engineering Department
University of Jordan
Amman 11942, JORDAN

ABSTRACT

In this work, we consider the problem of scheduling arriving jobs to a single machine where the objective is to minimize the mean tardiness. The scheduler has the option of reducing the processing time by half through the employment of an extra worker for an extra cost per job (setup cost). The scheduler can also choose from a number of dispatching rules. To find a good policy to be followed by the scheduler, we implemented a λ-SMART algorithm to do an on-line optimization for the studied system. The found policy is only optimal with respect to the state representation and set of actions available, however, we believe that the developed policies are easy to implement and would result in considerable savings as shown by the numerical experiments conducted.

1 INTRODUCTION

Reinforcement learning RL encompasses a variety of techniques that autonomous agents, which interact with their environment, use to select the action that achieves their goal of maximizing a certain reward. In a typical reinforcement-learning model, as shown in Figure 1, the agent senses the state of the environment *s*, performs an action *a* and receives a reward *r* due to the change in environment state or due to a direct reward such as minimizing tardiness in the model presented (Sutton and Barto 1998).

Reinforcement learning is found to be a better alternative to optimize Markov Decision Processes (MDP) and Semi-Markov Decision Processes (SMDP) than the classical Dynamic Programming (DP) approach (Sutton and Barto 1998). Online optimization, avoiding the curse of dimensionality and avoiding the curse of modeling are some of the reasons for choosing RL for optimizing stochastic systems (Gosavi 2004). A good survey paper about RL, its implementation and application is that of (Kaelbling, Littman and Moore 1996).

Different RL algorithms are suggested in the literature where the reader is referred to (Sutton and Barto 1998) for an excellent discussion about these algorithms. As the model implemented in this work deals with minimizing the average tardiness on the long run, a λ-SMART algorithm which emphasizes on the time taken for the state change is adopted in this work (Gosavi 2004). A good reference about RL techniques to minimize the average reward is found in (Mahadevan 1996).

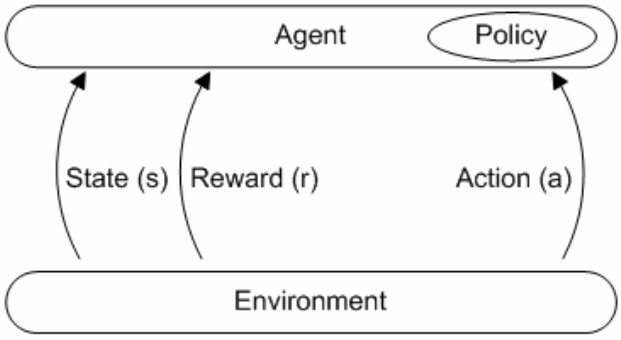

Figure 1: Agent-Environment Interaction in Reinforcement Learning System

The λ-SMART algorithm finds the optimal policy by finding the appropriate *Q(s,a)* values iteratively. The *Q(s,a)* values take into account the relation between the state s and action a. The *Q(s,a)* values are updated at each step that a reward is received for an action taken. The original Q-learning algorithm was first proposed by (Watkins 1989). The updating scheme of the Q-values is given by the following equation:

$$Q(s,a) = Q(s,a) + \alpha(r(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)), \quad (1)$$

where $Q(s,a)$ is the state-action pair value, s is the current state, a is the current action, s' is the next state, a' is the next action, $r(s,a)$ is the reward, $\gamma \in [0,1)$ is the discount factor of getting to a new state and α is the learning rate. The λ-SMART algorithm modifies Equation 1 by taking into account the average reward per time ρ for taking an action at a given state as explained in the algorithm presentation in Section 3.

The $Q(s,a)$ values represent the knowledge base of the agent which enables the agent to select the optimum action at each state and it solves the Bellman equations asynchronously. The λ-SMART algorithm allows the exploration of different actions at the different states.

2 MODEL

The system studied in this work consists of a single entity that might be considered as a single machine or a complete factory having a number of processes. Jobs of different processing requirements arrive to this machine and each has a due-date to satisfy where set-up times between jobs are neglected. The machine can only process one job at a time. We assume in this work that the processing time of any job can be reduced by half through the employment of a second worker or subcontracting part of the arriving job to an out of house manufacturer. The cost of hiring a new worker is denoted by K and its measuring unit is \$/job. The penalty of a tardy job is denoted by P and its measuring unit is \$/time.

The decision maker or the agent employed can choose whether to hire a new worker or not and which dispatching rule to implement. The available dispatching rules for the agent(s) are: SPT (Shortest Processing Time), EDD (Earliest Due Date), and FIFO (First In First Out). The agent can choose from a set of six actions which are the possible combinations of the dispatching rules and number of workers. In the following presentation, the number that follows the dispatching rule would indicated the number of workers employed such that; FIFO2 represents a FIFO dispatching rule where two workers are employed.

As stated earlier, the model developed by (Wang 2003) is partially adopted in this work which is explained in the experiment section. In his work, (Wang 2003) tried to find such appropriate dispatching rules without having the option of minimizing the processing time by employing an extra worker.

Based on the model described above, two objectives are considered in this work, namely; minimizing the tardiness of the finished jobs and minimizing the cost of hiring an extra worker. The model described is represented by Figure 2. The problem studied is not classical and to the authors' knowledge, no previous work is done with respect to the same cost function and decision variables.

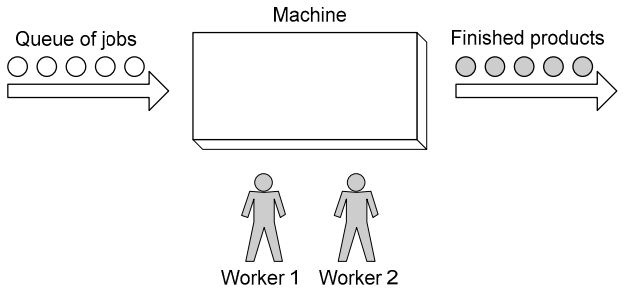

Figure 2: Model of the System

3 ALGORITHM

An appropriate representation of the system state would entail information about the number of jobs available at the time of taking an action, the processing time of each job and the due date of each job in the queue. As the number of possible combinations is tremendous, (Wang 2003) did replace this representation by measuring the lateness of the jobs in the system and then making this state space discrete. This representation was tried at the beginning of this work but did not result in a convergent policy with low operating cost.

The number of jobs in the queue is the state presentation employed in this work. We do believe that this state representation would not result in an optimal solution but did perform well when compared to the employment of a single dispatching rule. Another problem faced in this work was in evaluating the reward that the employed agent would get for the selected action at the selected state. The implemented rewarding scheme penalizes the use of the extra capacity in addition to the cost of the job being tardy as defined in the objective; so no positive reward is granted to the agent.

As actions taken might have future effects such as more jobs accumulating if a single worker is employed versus employing two workers, traces were added to the algorithm were an action might be penalized at later action- selection steps. These traces add the λ symbol to the λ-SMART algorithm (Gosavi, Bandla, and Das 2002). Finally, to guarantee convergence, a variable step size of $\alpha_n = 1.0/\sqrt{N+1.0}$ where N is the number of the leaving job is used as a learning rate for this algorithm. The algorithm can be presented through the following steps:

1. For $\forall s,a$, initialize $Q(s,a)=0$, $T=0$, $R=0$, and $N=0$.
2. For an idle machine with a queue of jobs available, choose an action (a) such that it has the maximum $Q(s,a)$ with probability 0.99, else randomly select another action.

3. Realize the new state (s'), the time needed to get to this state $(t(s')-t(s))$ and the reward achieved $r(t,s,a)$
4. If the action taken corresponds to the best Q value, then update

$$R \leftarrow R + r(s,a) \tag{2}$$

$$T \leftarrow T + t(s') - t(s) \tag{3}$$

and calculate

$$\rho = \frac{R}{T} \tag{4}$$

5. Calculate the temporal difference value TDV according to

$$TDV = r(s,a) + \rho(t(s') - t(s)) + \max_a Q(s',a) - Q(s,a) \tag{5}$$

6. For the selected action and the selected state, the trace value is incremented by 1, however; if the current state is the same as the future state then the value of the trace is made equal to 1 (Sutton and Barto 1998).
7. For $\forall s,a$, update the $Q(s,a)$ according to

$$Q(s,a) = \alpha_n * TDV * trace(s,a) \tag{6}$$

$$trace(s,a) \leftarrow trace(s,a) * 0.9 \tag{7}$$

8. increment N

$$N \leftarrow N + 1 \tag{8}$$

9. Repeat steps 2-8 until the departure of the 500,000th job.

The simulation model along with the optimization algorithm is coded using Java programming language, where a discrete event dynamic system DEDS model is implemented. The different jobs are represented by objects in the simulation model where each job has its own due date and processing time as described later. The jobs attributes are randomly and dynamically generated upon their arrivals. Only a single run is conducted to find the optimal policy then, another simulation run based on the best policy found is run to report the results presented in the next section.

4 NUMERICAL EXPERIMENT

The numerical example described is adopted from (Wang 2003); we briefly discuss it and explain our extension where a crushing cost per job is included. Jobs arriving to the single machine studied have an inter-arrival time of λ which is exponentially distributed and a mean processing time that is normally distributed where the mean is drawn from a uniform distribution having minimum and maximum values of a and b respectively and the standard deviation is 1/10 of the mean. A due date is assigned to each arriving job by multiplying an allowance factor by the processing time and adding it to the arrival time. The allowance factor has uniform distribution which is also uniform with parameters 1.2 and 1.8.

As explained earlier, we assume that there is a fixed cost paid per job to minimize its processing time by half. This strategy is always valid regardless of the dispatching rule employed which leaves the agent to choose one of six available actions at each state other than state 1. The other cost component that the agent tries to minimize is the tardiness cost where a penalty of $P = \$1/time$ is paid. The summation of both of these costs represents the total cost which is averaged over time during the learning process however; it is averaged over jobs when running the simulation based on the best policy found. Using an indicator function $1.\{\}$ to represent the selection of two workers when action a_i is taken, the cost equation is:

$$Cost = \frac{1.0}{500{,}000} \sum_{i=1}^{500{,}000} [P(C_i - d_i) + K1.\{a_i \in (SPT2, EDD2, FIFO2)\}] \tag{9}$$

The first columns of Table 1 show a comparison between the employment of four fixed policies in all states with respect to the policy that chooses the actions having the maximum *Q(s,a)* values at each state s. The *Q(s,a)* values are found by running the algorithm of the previous section while the results reported are again the average taken of a single simulation run where no warm-up period is considered and the simulation run ended with the departure of the 500,000th job.

The parameters employed in the first table for λ , a, b are 8, 6 and 8 respectively while different values of K are considered as shown in the table. The numbers within parenthesis show the corresponding cost of employing any of the policies. Any 1 symbol indicates using any dispatching rule employing 1 worker at state 1. It is seen that the worst cost is obtained by using SPT1 always and by paying a crushing cost of 5 \$/job, a significant decrease is obtained in the average cost paid. More savings are obtained by running the policy found by the λ-SMART algorithm.

The last column of Table 1 shows the results of employing the policies found by the λ-SMART algorithm (for *K=30*). It is worth noting that for *K=30* $/job, employing 2 workers is delayed till 3 jobs accumulate in the system's queue.

Table 1: Comparison between Different Cost Scenarios Using Various Dispatching Rules

s	*K=5* (-15.6)	*K=5* (-13.8)	*K=5* (-5.08)	*K=5* (-5.07)	*K=5* (-2.89)	*K=30* (-9.37)
1	SPT1	EDD1	SPT2	EDD2	Any1	Any1
2	SPT1	EDD1	SPT2	EDD2	EDD2	SPT1
3	SPT1	EDD1	SPT2	EDD2	EDD2	EDD2
4	SPT1	EDD1	SPT2	EDD2	EDD2	SPT2
5	SPT1	EDD1	SPT2	EDD2	FIFO2	SPT2
6	SPT1	EDD1	SPT2	EDD2	FIFO2	SPT2

Table 1 show that a state dependent policy performs better than using the same dispatching rule and number of workers always. EDD2 is the best policy to employ for low subcontracting cost where more than 1 job is present in the queue.

In the second experiment, the sensitivity of the policy and cost with respect to the problem parameters is studied for the case where *K=10* $/job. The three numbers preceding the costs are: λ , a, b respectively. As shown in Table 2, the best rule to implement once jobs start queuing is EDD2, this rule remained the same for the different cases presented in Table 2. The costs found reflect the congestion faced by the system.

Table 2: Effect of Changing Inter-Arrival Time and Processing Time on the Policy Selected

s	8,(6,8) (-4.54)	8,(8,10) (-7.05)	10,(6,8) (-3.15)	10,(8,10) (-5.13)
1	Any1	Any1	Any1	Any1
2	EDD2	EDD2	EDD2	EDD2
3	EDD2	EDD2	EDD2	EDD2
4	SPT2	EDD2	EDD2	EDD2
5	SPT2	EDD2	SPT2	EDD2
6	SPT2	EDD2	SPT2	SPT2

5 CONCLUSION AND FUTURE WORK

Despite the misrepresentation of the system state, the results obtained showed a significant savings in the overall systems cost as the first three columns in Table 1 show. It is found that employing 2 workers is an efficient choice for states having more than 1 job in the system, however, if the cost of hiring a new worker gets high, the system would delay the hiring of a second worker until more than 2 jobs are present in the system as shown in the case where K=$30/job in Table 1. Using EDD2 rule seems to be a good policy to employ if the jobs congest in the system and the job controller has the choice to employ a new worker to reduce the processing time.

Currently, our group is trying to use a different representation of the system state. Some of the options considered for this representation are; congestion measures such as those suggested by (Lee, Bhaskaran and Pinedo 1997) or defining the system state based on the average tardiness found by a dispatching rule. One modification to the system might be the use of a different cost function to reduce the processing time as done usually in real life situations (Pinedo 2001). Using other dispatching rules might also be a different modification to the model studied.

Extending the system architecture to a flow shop or job shop where more than one worker are employed would make the system more complex and challenging to optimize. Issues such as work force flexibility might be considered in such systems.

REFERENCES

Gosavi, A. 2004. Reinforcement learning for long-run average cost, European Journal of Operational Research, vol. 155: 654-674.

Gosavi, A., N. Bandla, and T. Das. 2002. A reinforcement learning approach to airline seat allocation for multiple fare classes with overbooking, IIE transactions, 34 (9), 729-742.

Kaelbling, L., M. Littman, and A. Moore. 1996. Reinforcement Learning, a survey, Journal of Artificial Intelligence Research vol. 4, pp 237-285.

Lee, Y. H., K. Bhaskaran, and M. Pinedo. 1997. A heuristic to minimize the total weighted tardiness with sequence-dependent setups, IIE transactions, 29:45-52.

Mahadevan, S. 1996. Average reward reinforcement learning: foundations, algorithms, and empirical results, Machine Learning 22(1): 159-195.

Pinedo, M. 2001. Scheduling: Theory, Algorithms and Systems, Prentice Hall, 2nd edition.

Sutton, R., and A. G. Barto. 1998. Reinforcement Learning, The MIT press, Cambridge, Massachusetts.

Wang, Yi-Chi. 2003. Application of reinforcement learning to multi-agent production system, Ph.D. dissertation, Department of Industrial Engineering, Mississippi State University.

Watkins, C. J. 1989. Learning from delayed rewards, Ph.D. thesis, Kings College, Cambridge, England.

AUTHOR BIOGRAPHIES

HADEEL D. IDREES is a student pursuing her bachelor degree in Industrial Engineering at the University of Jordan. Her e-mail address is <hadeel.idrees@gmail.com>.

MAHDY O. SINNOKROT is a student pursuing his bachelor degree in Industrial Engineering at the University of Jordan.. His e-mail address is <msinnokrot@gmail.com>.

SAMEH AL-SHIHABI is Assistant Professor in the Department of Industrial Engineering at the University of Jordan. His research interests include simulation optimization and combinatorial optimization. His e-mail address is <s.shihabi@ju.edu.jo>.