Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

A Case Study in Scheduling Reentrant Manufacturing Lines: Optimal and Simulation-Based Approaches

José A. Ramírez-Hernández and Emmanuel Fernandez

Abstract— **This paper presents initial results of a research study in the optimal scheduling (i.e., job sequencing) in Reentrant Manufacturing Lines (RML), motivated by applications in semiconductor manufacturing. In particular, a simple benchmark RML is utilized, and the optimal scheduling policy is analyzed for an infinite horizon discounted cost problem formulation. The optimality equation and condition are derived, and optimal policy results are obtained for general non-negative one-stage cost functions (in the buffer size). Computational experiments are also performed using the Modified Policy Iteration algorithm. Preliminary experiments on the application of a Neuro-Dynamic Programming (NDP) method (i.e., Q-learning) to approximate the optimal scheduling policy are then presented, when linear and quadratic one-stage cost functions are considered. These experiments show that the Q-learning algorithm gradually approximates the optimal policy as the number of iterations increases and longer simulation lengths are utilized. However, the computational load required by the algorithm increases exponentially with the number of states. Results from this study represent an initial and exploratory research in the application of NDP methods to large-scale RML systems. More extensive research in both exact optimal results and efficient NDP schemes is in progress.**

I. INTRODUCTION

For the last two decades, the rapid development and economic impact of the semiconductor manufacturing industry has substantially increased the amount of research conducted on the analysis, modeling, and control of Reentrant Manufacturing Lines (RML) [1], [2], [3], [4], [5]. RML are systems in which the parts being fabricated return to one or more work stations (e.g., group of tools) on a repeated basis. This body of research has been justified by the complexity associated with these systems. For instance, in the fabrication of Very Large Scale Integration (VLSI) electronic circuits, the number of processing steps required to obtain a finished device can be easily counted in the hundreds [6]. In terms of modeling, analysis, and control, such complexities are represented by large state spaces and intractable mathematical models, and, therefore, the difficulties to design optimal or near to optimal control policies are increased [7].

The main control problem in RML, and its application to semiconductor manufacturing systems (SMS), is known as *Floor Shop Control* (FSC) [8]. It focuses on scheduling problems [5], which are also categorized as: *job sequencing* and *input regulation*. In the former task, decisions are made to select which lot of material will be served next when two or more lots are waiting, and a tool or machine is available to receive work. In the latter, the decisions are either release or not a new lot into the system at a given time. Different approaches have been studied for FSC in SMS [8], [1], [9], most of them based on heuristics, control theory, and artificial intelligence.

The control of some RMLs is a difficult task because of the complexity of such systems. For instance, realistic RML models for SMS are composed of dozens of queues and hundreds of steps in the manufacturing process [5]. Thus, analytical model complexity leads to intractable exact optimal control solutions, except for simple benchmark problems. However, it is a widespread practice in the semiconductor manufacturing industry to develop and maintain sophisticated simulation models to assess the impact of operations in overall performance. This may facilitate the successful application of simulation-based optimization methods such as Neuro-Dynamic Programming (NDP) [10].

The purpose of this paper is then to address both analytical and simulation-based approaches in the optimization of RML scheduling and its possible application to SMS. The work presented here is motivated in part by previous research in optimal scheduling and capacity allocation in semiconductor manufacturing conducted by the authors and others [11], [12], [13]. The first objective of this paper is to present the optimality equations and condition for the scheduling (i.e., job sequencing) of a simple benchmark RML, under an infinite horizon discounted cost (DC) formulation [14], [15] and under general non-negative one-stage cost functions (in the buffer size). In this paper, the DC criterion was selected considering the current dynamics in the semiconductor industry, where costs and benefits in the short-term are important (e.g., during ramp-up of new products [16]) given the short life-cycle of semiconductor devices [17]. Although the RML model presented in this paper is quite simple, the results obtained from this exercise are valuable to study and understand the basic difficulties associated with the synthesis of optimal (or near to optimal) scheduling policies in RML for SMS.

The second objective is to present initial results of our exploratory research in the application of the NDP methods for the approximation of optimal RML scheduling policies, specifically, by using the Q-learning algorithm [10], [18], [19], [20]. In this case, our objective was to assess the feasibility of applying NDP by utilizing a benchmark model with a known optimal policy, and later on study its scalability to more complex models for which optimal solutions may not be known. Therefore, we present results of the effect of varying the parameters of Q-learning on the performance

José A. Ramírez-Hernández and Emmanuel Fernandez are with the Department of Electrical & Computer Engineering & Computer Science, University of Cincinnati, OH 45221, USA. Emails:{ramirejs;emmanuel}@ececs.uc.edu.

0-7803-9568-9/05/$20.00 ©2005 IEEE

obtained by policies derived by utilizing this algorithm.

This paper is organized as follows: in section II, we present the general model for the benchmark RML. The optimization model using an infinite horizon discounted cost is presented in section III. The main analytical results are presented in section IV, followed by the numerical examples of optimal policies obtained through the MPI algorithm in section V. The application of Q-learning to approximate optimal scheduling policies is discussed in section VI. Conclusions are presented in section VII.

II. BENCHMARK REENTRANT MANUFACTURING LINE MODEL

The model of the benchmark RML discussed in this paper has been previously presented in e.g., [21], [5], [22]. As shown in Figure 1, the RML model consists of a manufacturing line with three buffers and two machines.

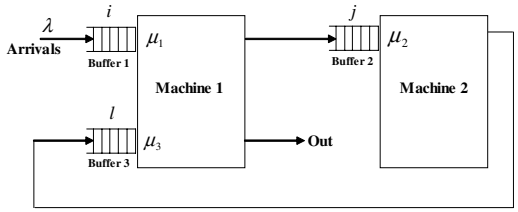

Fig. 1. Benchmark Reentrant Manufacturing Line.

The state of the system is represented as the tuple $s(t) \equiv (i(t), j(t), l(t))$ corresponding to the buffer levels at time *t,* with $s(t) \in S$, and $S := \{(i,j,l)|i,j,l \in \mathbb{Z}^*\}$ as the state space. For this system, control decisions deal with in job scheduling in the manufacturing line. In other words, the control system needs to select between Buffer 1 or Buffer 3 to be served next in Machine 1. Therefore, the set of control actions is defined by $U := \{0,1\}$, where $u = 1$ when the action is *serve Buffer 1*, and $u = 0$ when the control is *serve Buffer 3*. Thus, the system corresponds to a Semi-Markov Decision Process (SMDP) with a continuous-time Markov chain where the state is given by the tuple $s(t)$.

This system has a simple production sequence: new part arrival→Buffer 1, Machine 1→Buffer 2, Machine 2→Buffer 3, Machine 1→Exit. The processing times at each machine are exponentially distributed with means $\frac{1}{\mu_1}$, $\frac{1}{\mu_2}$, and $\frac{1}{\mu_3} = \frac{1}{\mu_1}$, respectively. The arrival of lots of material are also exponentially distributed with mean $\frac{1}{\lambda}$.

III. OPTIMIZATION MODEL: INFINITE HORIZON DISCOUNTED COST

The optimization model considers the minimization of an infinite horizon discounted cost as follows:

$$\lim_{N\to\infty} \; E\left\{\int_0^{t_N} e^{-\beta t} g\left(s(t),u(t)\right) dt\right\}, \tag{1}$$

where β is a positive scalar, and $g\left(s(t),u(t)\right)$ is a continuous-time cost which is a function of the state $s(t)$ and the control action $u(t) \in U$ at time *t.* In order to obtain a discrete-time optimization model, a discrete, and statistically equivalent, version of the continuous-time Markov chain of the system presented in section 2 is obtained through the procedure of *uniformization* [23], [15]. The resulting Bellman's equation has the following form:

$$J(s) = \min_{u\in U(s)} E\left\{\tilde{g}\left(s,u\right) + \alpha J(f(s,u))\right\}, \tag{2}$$

where $\tilde{g}\left(s,u\right)$ is the one-stage cost, and α is a discount factor, with $0 < \alpha < 1$. The next state function is given by $f(s,u)$, with $s \in S$, and $u \in U$. For the discrete-time system, the discount factor and the one-stage cost are defined as follows [15]:

$$\alpha = \frac{\nu}{\beta+\nu}, \tag{3}$$

$$\tilde{g}\left(s,u\right) = \frac{g\left(s,u\right)}{\beta+\nu}, \tag{4}$$

where ν is the uniform transition rate, with $\nu \geq \nu_s(u)$ for all $s \in S$, $u \in U$; and $\nu_s(u)$ is the transition rate at the state s given the control u. For the benchmark RML system, ν is defined as follows:

$$\nu = \lambda + \mu_1 + \mu_2 + \mu_3. \tag{5}$$

We follow the state transition equations representation presented in [5], which are based on the uniformization procedure. Thus, let $\{\tau_n\}$ a sequence of times where the continuous-time Markov chain is sampled with $\tau_0 = 0$. Each time instant where the system change its state is considered a sample time in the uniformized version. It is also assumed that the control policy does not change during the interval $[\tau_n, \tau_{n+1})$. These type of policies are defined as *non-interruptive* [5]. Similarly, the control design is limited to *non-idling* policies [5], [22] where a tool or machine is not permitted to remain idle if at least one buffer has one or more lots to be processed. Finally, if $s(\tau_n) = (i,j,l)$, then the state transition equations are as follows [5]:

$$s(\tau_{n+1}) = \begin{cases} (i+1,j,l) & \text{w.p } \frac{\lambda}{\nu} \\ ((i-1)^+, j+\mathbb{I}(i), l) & \text{w.p. } \frac{\mu_1}{\nu} \\ & \text{if } u=1 \\ (i,(j-1)^+, l+\mathbb{I}(j)) & \text{w.p. } \frac{\mu_2}{\nu} \\ & \text{if } u=1 \text{ or } 0 \\ (i,j,(l-1)^+) & \text{w.p. } \frac{\mu_3}{\nu} \\ & \text{if } u=0 \\ (i,j,l) & \text{otherwise,} \end{cases} \tag{6}$$

where $(\bullet)^+ = \max(0,\bullet)$, and $\mathbb{I}(\delta) = \begin{cases} 1 \text{ if } \delta > 0 \\ 0 \text{ if } \delta \leq 0 \end{cases}$.

Thus, the Bellman's equation can be expressed in the following way:

$$J(s) = \frac{1}{\beta+\nu} \min_{u\in U(s)} [g\left(s,u\right) + \nu \sum_{s'} \tilde{p}(u)_{ss'} J(f(s,u))], \tag{7}$$

where $s, s' \in S$, and s' represents the next state, with $s' = s(\tau_{n+1}) = f(s,u)$, and $\tilde{p}(u)_{ss'} = P\{s'|s,u\}$ is the

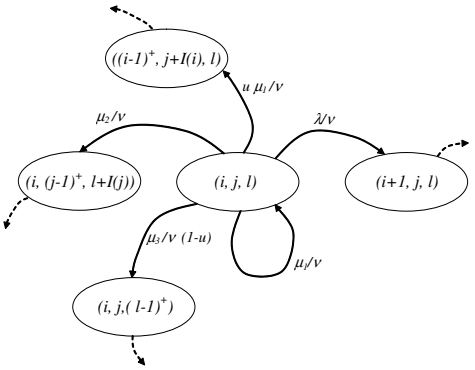

Fig. 2. State transitions diagram for the benchmark RML system.

conditional transition probability for the uniform version of the continuous-time Markov chain. Figure 2 depicts the state transition diagram of the benchmark RML.

IV. OPTIMAL SCHEDULING POLICY

This section presents the main result of this paper, which is an optimal scheduling (i.e., job sequecing) solution for the benchmark Reentrant Manufacturing Line (RML). This RML was also utilized by Suk and Cassandras [21] to derive an optimal solution in the case of buffers with finite capacity and for a non-linear one-stage cost. Here we present the case of infinite capacity and general non-negative one-state cost functions. Before presenting the results, however, we provide several assumptions, definitions, and some notation.

A. Assumptions, definitions, and notation

The following assumptions are made for the scheduling problem of the benchmark RML.

Assumption 1: The one-stage cost function $g(s,u)$ is assumed to depend only on the state $s \in S$. Then $g(s,u) \equiv g(s)$.

Definition 1: The componentwise partial order on S, denoted "$\leq_{cw}$", is defined as follows: for any $v = (v_1, v_2, v_3)$, $w = (w_1, w_2, w_3) \in S$, we say that $w \leq_{cw} v$ iff $w_q \leq v_q$ for q=1, 2, 3; and for all $w, v \in S$.

Consider $h : S \to \mathbb{R}$, such that $h(s) \geq 0 \ \forall \ s \in S$. If $h(w) \leq_{cw} h(v)$ for any $w, v \in S$, $w \leq_{cw} v$, then $h(\cdot)$ is said to be non-negative and monotonically nondecreasing with respect to the componentwise partial order "$\leq_{cw}$".

Assumption 2: The one-stage cost function $g(s)$ is a non-negative function and monotonically nondecreasing with respect to the usual componentwise partial order "$\leq_{cw}$" for all $s \in S$.

Assumption 2 is natural for the scheduling problem in the benchmark RML; that is, it is expected to obtain higher costs as the work in process is increased at each buffer of the system. We also assume that work starvation is avoided in the benchmark RML; therefore, there is a set of states for which a *non-idling stationary policy* is applied. For the benchmark RML system, this policy is defined as follows:

Definition 2: In the benchmark RML, a stationary scheduling policy $\pi_{NI} = \{u, u, u, ...\}$, where $u(s) : S \to U$, is non-idling if
$u = 1$ for all $i > 0$, $l = 0$, and $j \geq 0$,
$u = 0$ for all $l > 0$, $i = 0$, and $j \geq 0$, and
$u = 0$ for all $i = 0$, $l = 0$, and $j \geq 0$.
In addition, the set of states for which the non-idling conditions are applied is defined as S_{NI}. Therefore, the subset of states for which a decision has to be made from the Bellman's optimality equation is defined as

$$S_{\overline{NI}} := \{(i,i,j) \in S | i > 0, j \geq 0, l > 0\}, \ \ S_{\overline{NI}} \subseteq S. \quad (8)$$

The following notation is utilized throughout this paper: let $J(i,\, j,\, l) \equiv J(s)$ and $g(i,\, j,\, l) \equiv g(s)$, where the system's state is represented by the tuple $(i,\, j,\, l)$. The Bellman's optimality equation for the scheduling problem of the RML can then be rewritten as follows:

$$\begin{aligned} J(i,\, j,\, l) = \tfrac{1}{\beta+\nu} \, [g(i,\, j,\, l) + \lambda \, J(i+1,\, j,\, l) + \\ \mu_2 J(i,\, (j-1)^+,\, l + \mathbb{I}(j)) + \mu_3 \, J(i,\, j,\, (l-1)^+) + \\ \mu_1 J(i,\, j,\, l) + \mu_1 \text{min}_u \, \{u \cdot \Delta(i,\, j,\, l)\}] \, , \\ \forall \ (i,\, j,\, l) \in S, \end{aligned} \quad (9)$$

where,

$$\begin{aligned} \Delta(i,\, j,\, l) = J((i-1)^+,\, j + \mathbb{I}(i),\, l) - J(i,\, j,\, (l-1)^+) \\ \forall \ (i,\, j,\, l) \in S. \end{aligned} \quad (10)$$

Finally, the following definition will be utilized in the presentation of several results in the next subsections.

Definition 3: Let $(i,j,l), (i',j',l') \in S$, where $i' \geq i$, $j \geq j'$, and $l \geq l'$; with corresponding costs $J(i,j,l)$ and $J(i',j',l')$. Then

$$\Delta'(i,j,l,i',j',l') \equiv J(i,j,l) - J(i',j',l'). \quad (11)$$

B. Results

From the previous subsection and (9), the general optimality condition below follows immediately.

Theorem 1: For the scheduling problem of the benchmark RML, if $(i,j,l) \in S$, then it is optimal to serve buffer 3 iff the state $(i,j,l) \in S_3 \subseteq S$, where

$$S_3 = \{ (i,\, j,\, l) \in S | \, \Delta(i,\, j,\, l) \geq 0 \} \, .$$

Otherwise, it is optimal to serve buffer 1.

Lemma 1: For the scheduling problem of the bechmark RML, let $(i,j,l), (i',j',l') \in S$, where $i' \geq i$, $j \geq j'$, $l \geq l'$, and let $\Delta'(i,j,l,i',j',l')$ given in Definition 3. Then, $\Delta'(i,j,l,i',j',l') \geq 0 \ \forall \ (i,j,l), \ (i',j',l') \in S$ if

$$g(i,j,l) - g(i',j',l') \geq 0, \quad (12)$$

otherwise $\Delta'(i,j,l,i',j',l') < 0$.

Proof: The proof is given by induction and value iteration. Due to space restrictions is not included here, see [24]. ■

Theorem 2: For the scheduling problem of the benchmark RML it is optimal to serve buffer 3 if

$$g((i-1)^+, j + \mathbb{I}(i), l) - g(i,j,(l-1)^+) \geq 0, \ \ (i,j,l) \in S,$$

otherwise it is optimal to serve buffer 1.

Proof: By considering that

$$\Delta(i,j,l) = \Delta'((i-1)^+, j+\mathbb{I}(i), l, i, j, (l-1)^+),$$

the proof of Theorem 2 follows immediately from Theorem 1 and Lemma 1. ■

Intuitively, we can interpret the results in Theorem 2 as follows: at each state transition in the benchmark RML it is optimal to serve the buffer that generates the next state with the smaller one-stage cost, i.e., the optimal policy exhibits a myopic behavior.

Corollary 1: Let $(i,j,l) \in S_{\overline{NI}} \subseteq S$, then it is optimal to serve buffer 3 if

$$g(i-1, j+1, l) - g(i,j,l-1) \geq 0,$$

otherwise it is optimal to serve buffer 1.

C. Examples

The following are two simple examples where a linear and a quadratic one-stage are utilized.

Example 1: Let $g(i,j,l) = i+j+l$ and the state is in $S_{\overline{NI}}$. Thus, by applying Theorem 2 and Corollary 1, it is optimal to serve buffer 3 $\forall\ s \in S_{\overline{NI}}$ since

$$g(i-1, j+1, l) - g(i,j,l-1) = 1 > 0.$$

This policy is identical to the so-called Shortest Processing Time (SPT) policy [1], [9] where the parts or lots with smaller processing times ahead receive a higher priority to being served.

Example 2: Let $g(i,j,l) = i^2+j^2+l^2$, and the state is in $S_{\overline{NI}}$. Thus, from Theorem 2 and Corollary 1, it is optimal to serve buffer 3 if and only if

$$j + l + \frac{1}{2} \geq i \Leftrightarrow j + l \geq i, \text{ with } i,j,l \in \mathbb{Z}^*. \qquad (13)$$

Numerical solutions for examples 1 and 2 are described in the next section, where the MPI algorithm is utilized to compute the optimal policy.

V. NUMERICAL SOLUTION BY APPLYING THE MODIFIED POLICY ITERATION ALGORITHM

This section presents the numerical solution for the optimal scheduling policy when a linear and a quadratic one-stage cost are utilized. Our objective was to verify the result from Theorem 2 by utilizing the MPI algorithm [14], [15] to approximate the optimal scheduling policy. This algorithm was implemented in Matlab 6.5.

The MPI algorithm was applied to the optimization model of the benchmark RML system by using the two one-stage cost functions presented in the examples 1 and 2: $g_1(s) = i+j+l$, and $g_2(s) = i^2+j^2+l^2$.

The following are the general conditions utilized to compute the scheduling policy by using the MPI algorithm:

- The system parameters were selected as presented by Chen & Meyn in [22]: $\mu_1 = \mu_3 = 0.3492$, $\mu_2 = 0.1587$, and $\lambda = 0.1429$.
- For the parameters selected, the system is stable under any non-idling policy [4], [22], [5].
- Cost parameter: $\beta = 0.7 \Rightarrow$ α=0.588.
- Buffer sizes are arbitrarily limited to 80 lots in buffers 1 and 2, and 50 lots in buffer 3; therefore, the resulting space state has 80x80x50=320000 different states.
- The initial cost vector for the MPI algorithm was set to zero, and the initial policy was *Serve Buffer 1*, for all $s \in S$.

The policy resulting from applying the MPI algorithm when the quadratic one-stage cost was utilized is depicted in Figure 3. When a linear cost was considered, the optimal policy converged to the policy presented in example 1. The MPI algorithm required a minimum of 10 iterations to converge when the discount factor was β=0.7.

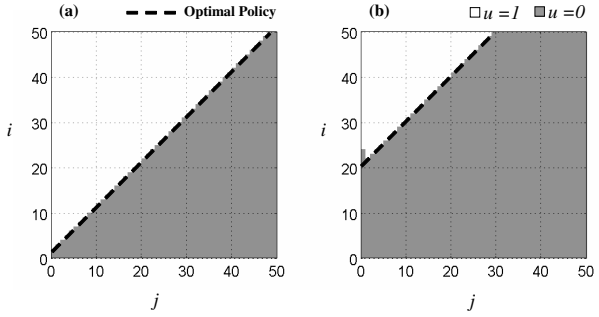

Fig. 3. Scheduling policy obtained with the MPI algorithm for the quadratic one-stage cost function $g_2(s)$ and β = 0.7, with $i,\ j \geq 0$ and: (a) $l = 1$, (b) $l = 20$. Dashed line represents the optimal policy given in (13).

As expected, the optimal policies resulting from applying the MPI algorithm verified the result from Theorem 2 and examples 1 and 2 in section IV.

VI. NEURO-DYNAMIC APPROACH TO APPROXIMATE THE OPTIMAL SCHEDULING POLICY

In this section we present the results of both numerical and simulations experiments of the application of a NDP method [10], [18], [19] in the approximation of the optimal scheduling policy for the benchmark RML. Our objective was therefore to assess the feasibility of the application of NDP to scheduling in RML, with the objective of subsequently exploring the scalability of these methods to more complex models for which optimal solutions may not be known.

In the experiments presented in this section, we used the Q-learning algorithm [20]. Although Q-learning can be considered one of the most basic, but well known methods of NDP, this method was selected because it was easily implementable and solid convergence results are available [10].

An approximation to the optimal policy π^* is then obtained with the Q-learning algorithm by computing the Q-factors $Q(s,u)$ [10], [18], [20], by using a simulation model, and by utilizing the following iterative equation:

$$Q_{t+1}(s,u) = (1-\gamma_t(s,u))Q_t(s,u) + \gamma_t(s,u)\left(g(s,u,s') + \alpha \min_v Q_t(s',v)\right), \qquad (14)$$

where s' is the next state and $\gamma_t(s,u)$ is a non-negative step-size that is reduced to zero as t$\rightarrow\infty$ in order to obtain convergence to the optimal Q-factors $Q^*(s,u)$ [10].

A. Numerical example: learning an optimal policy with NDP

For this numerical example we considered the quadratic one-stage cost function from example 2 in section IV. The implementation of the Q-learning algorithm was performed with the simulation software ARENA version 7.01 [25], and by coding the algorithm in Visual Basic. The following are some of the general conditions considered in the experiment:

- The parameters for the benchmark RML were the same as those utilized in the numerical example of section V.
- In order to update the Q-factors, multiple replications were performed.
- To avoid overhead in the simulator, the maximum number of lots allowed in the system was set to 100 lots maximum per buffer. Thus, the implementation required a lookup table of $100^3\times$ 2 = 2 million entries.
- During the learning phase, *exploration* [10], [18] was included in the selection of the control action by using the so-called ϵ-*greedy* policy [18]. As suggested in previous numerical experiments presented in [18], we selected $\epsilon = 0.1$.
- The discount factor was set to $\beta = 0.7 \Longrightarrow \alpha$=0.588 to provide a relatively faster convergence of the algorithm. In addition, all Q-factors are initialized at zero.

Figures 4(a)-(f) illustrate how the optimal scheduling policy is gradually approximated by the Q-learning algorithm as the number of iterations was increased (i.e., number of state-control visits is increased). However, the approximated policy remains far from the optimal policy depicted in Figure 3. This can be attributed to the large number of Q-factors (2 million) that slow-down the convergence to the optimal policy due to the increased computational load (and memory) required during the learning process.

B. Simulation experiments: performance evaluation of near to optimal policies obtained by NDP

In this subsection we present simulation experiments that were utilized to evaluate the performance (i.e., discounted cost) of different scheduling policies for the benchmark RML obtained through the Q-learning algorithm. With these experiments we also compared the effect in the performance of the approximated policy by varying the different parameters in the Q-learning algorithm: number of simulation replications N utilized to approximate the optimal policy; simulation length L of each replication; and exploration rate $\epsilon \in [0,1]$ utilized by the ϵ-greedy policy to select the control action during the learning process.

Each experiment consisted of two parts: first, an approximation to the optimal policy was performed by using Q-learning; and second, the approximated policy was evaluated using the simulation model and by computing an estimation of the expected discounted cost $J^\pi(s)$ (i.e., performance index) under such policy. These statistics were computed by simulation with 400 replications of 300 time units. The

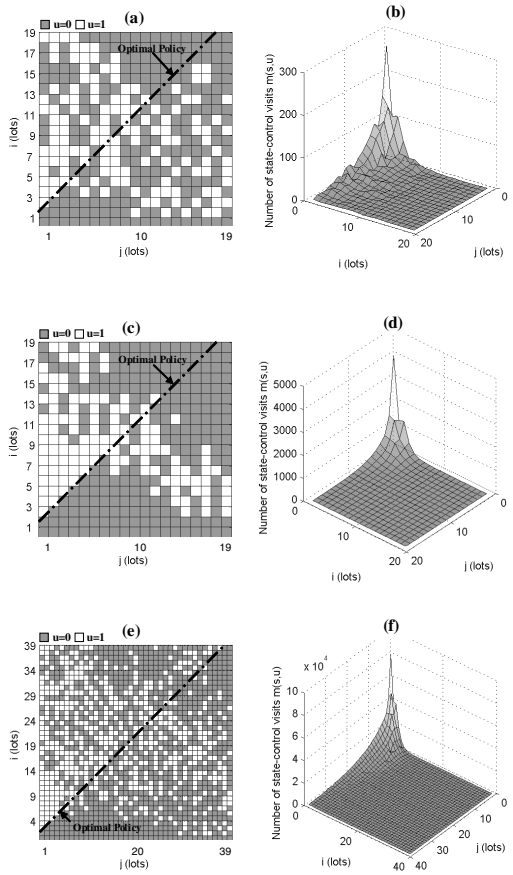

Fig. 4. Optimal scheduling policy approximation by Q-learning with a quadratic one-stage cost function and for $i \geq 0$, $j \geq 0, l = 1$: (a)-(d) 100 and 32000 iterations simulated for 5000 units of time, respectively; and (e)-(f) 33000 iterations simulated for 60000 units of time. While the figures on the left show the policy approximation, those on the right show the number of state-control visits $m_t(s,u)$ during the experiment.

discount factor β was set to 0.1 to provide a slower rate of convergence. However, notice from Theorem 2 that the optimal policy does not depends on β.

As a reference to compare the performance of the approximations to the optimal policies, we evaluated the optimal policies obtained in examples 1 and 2 in the simulation model. Table I shows these results with the corresponding 95% half-length confidence intervals (CI).

TABLE I
COMPUTED OPTIMAL DISCOUNTED COST FOR LINEAR AND QUADRATIC ONE-STAGE COST FUNCTIONS

Policy	$J^\pi(s)$
π_{LC}	18.23±0.65
π_{QC}	36.15±2.67

π_{LC}, π_{QC} : optimal policies when a linear and quadratic one-stage cost is considered, respectively.

Figure 5 shows the performance obtained by the approximations to the optimal policy given by the Q-learning

algorithm under different combinations of parameters in the learning algorithm. On the one hand, when a linear one-stage cost function was considered (Figures 5(a)-(b)), and given that the Q-factors were initialized in zero, the Q-learning algorithm started with an initial policy that is the optimal (see example 1). Therefore the performance obtained remained close to the optimal. On the other hand, when a quadratic one-stage cost was utilized and the Q-factors were initialized at zero, the initial policy was not the optimal. Thus, in this case it seems that there is a tendency to better approximate the optimal performance as the number of replications N is increased and longer simulation lengths are utilized (see Figures 5(c)-(d)). However, the confidence intervals are not statistically different except for $N = 1$ in Figures 5(c)-(d).

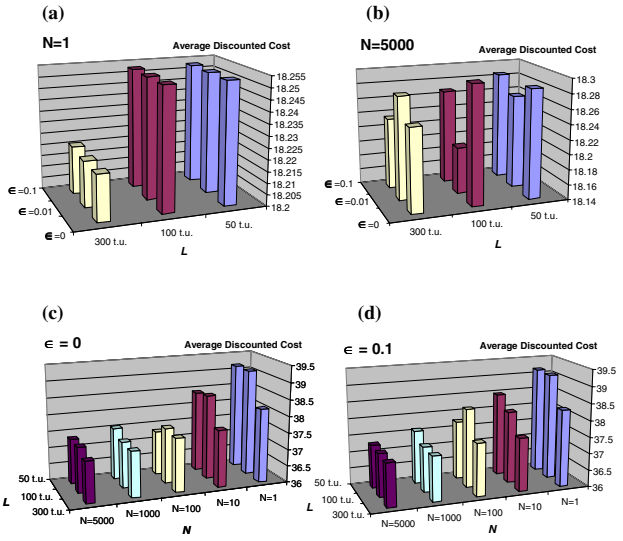

Fig. 5. Performance of approximations to the optimal policy for different combinations of number of replications N, simulation length L (time units), and exploration rate ϵ during the learning process: (a)-(b) approximation to π_{LC} (95% CI: ± 0.65), (c)-(d) approximation to π_{QC} (95% CI: ± 2.67).

VII. CONCLUSION

This paper presented initial results of a research study on analytical and simulation-based optimization (i.e., NDP) of job sequencing of RML. A simple benchmark RML was utilized to obtain an optimal scheduling policy for the infinite horizon discounted cost problem and for a general non-negative one-stage cost function. In addition, we presented preliminary experiments on the application of the NDP method Q-learning to approximate the optimal scheduling policy for two different one-stage cost functions: linear and quadratic. These experiments showed that the Q-learning algorithm can gradually approximate the optimal policy. However, the computational load required by the algorithm increases exponentially with the number of states; therefore, this could result restrictive if realistic RML are considered. Thus, our current research is focused on the study of other NDP methods based on compact state and action spaces representations.

REFERENCES

[1] L. Wein, "Scheduling semiconductor wafer fabrication," *IEEE Transactions on Semiconductor Manufacturing*, vol. 1, pp. 115–130, 1988.

[2] P. Kumar, "Re-entrant lines," *Queueing Systems: Theory and Applications*, vol. 13, pp. 87–110, 1993.

[3] C. Lu, D. Ramaswamy, and P. Kumar, "Efficient scheduling policies to reduce mean and variance of cycle-time in semiconductor manufacturing plants," *IEEE Transactions on Semiconductor Manufacturing*, vol. 7, no. 3, pp. 374–388, 1994.

[4] P. Kumar and S. Meyn, "Stability of queueing networks and scheduling policies," *IEEE Transactions on Automatic Control*, vol. 40, no. 2, pp. 251–260, 1995.

[5] S. Kumar and P. Kumar, "Queueing network models in the design and analysis of semiconductor wafer fabs," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 5, pp. 548–561, 2001.

[6] J. Plummer, M. Deal, and P. Griffin, *Silicon VLSI Technology*. Englenwood Cliffs, N. J.: Prentice-Hall, 2000.

[7] C. Papadimitriou and J. Tsitsiklis, "The complexity of optimal queueing network control," *Mathematics of Operations Research*, vol. 24, no. 2, pp. 293–305, 1999.

[8] R. Uzsoy, C. Lee, and L. Martin-Vega, "A review of production planning and scheduling models in the semiconductor industry part II: Shop-floor control," *IIE Transactions*, vol. 26, no. 6, pp. 44–55, 1994.

[9] S. Panwalker and W. Iskander, "A survey of scheduling rules," *Operation Research*, vol. 25, pp. 45–61, 1977.

[10] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Bellmont, MA: Athena Scientific, 1996.

[11] X. Yao, E. Fernández-Gaucherand, M. Fu, and S. Marcus, "Optimal preventive maintenance scheduling in semiconductor manufacturing," *IEEE Transactions on Semiconductor Manufacturing*, vol. 17, no. 3, pp. 345–365, 2004.

[12] J. A. Ramírez-Hernández and E. Fernández-Gaucherand, "An algorithm to convert wafer to calendar-based preventive maintenance schedules for semiconductor manufacturing systems," in *in Proceedings of the 42nd IEEE Conference on Decision and Control*, December 2003, pp. 5926–5931.

[13] S. Bhatnagar, E. Fernandez-Gaucherand, M. C. Fu, Y. He, and S. I. Marcus, "A markov decision process model for capacity expansion and allocation," in *in Proc. 38th IEEE Conference on Decision and Control*, December 1999, pp. 121–125.

[14] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY: John Wiley & Sons, Inc, 1994, ch. Discounted Markov Decision Problems, pp. 142–266.

[15] D. Bertsekas, *Dynamic Programming and Optimal Control*, 2nd ed. Bellmont, MA: Athena Scientific, 2000, vol. II.

[16] R. Sturm, J. Dorner, K. Reddig, and J. Seidelmann, "Simulation-based evaluation of the ramp-up behavior of waferfabs," in *Advanced Semiconductor Manufacturing Conference and Workshop, 2003 IEEEI/SEMI*, 31 March-1 April 2003, pp. 111–117.

[17] Semiconductor Industry Association (SIA), EECA, JEITA, KSIA, and TSIA. (2003) International technology road map for semiconductors (ITRS) 2003. [Online]. Available: http://public.itrs.net

[18] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[19] A. Gosavi, *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning*, 1st ed. Norwell, MA: Kluwer Academic Publishers, 2003.

[20] C. Watkins, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[21] J.-B. Suk and C. Cassandras, "Optimal control of a storage-retrieval queuing system," in *Proceedings of the 28th IEEE Conference on Decision and Control*, 1989, pp. 1093–1098.

[22] R.-R. Chen and S. Meyn, "Value iteration and optimization of multiclass queueing networks," *Queueing Systems*, vol. 32, pp. 65–97, 1999.

[23] S. Lippman, "Applying a new device in the optimization of exponential queuing systems," *Operations Research*, vol. 23, pp. 687–710, 1975.

[24] J. A. Ramírez-Hernández and E. Fernandez, "Optimal and simulation-based approaches for scheduling of a reentrant manufacturing line," 2005, manuscript in preparation, University of Cincinnati.

[25] W. Kelton, R. Sadowski, and D. Sturrock, *Simulation with Arena*, 3rd ed. USA: McGraw-Hill, 2003.