Proceedings of the 2007 IEEE Symposium on Approximate
Dynamic Programming and Reinforcement Learning (ADPRL 2007)

An Approximate Dynamic Programming Approach for Job Releasing and Sequencing in a Reentrant Manufacturing Line

José A. Ramírez-Hernández and Emmanuel Fernandez

Abstract— **This paper presents the application of an approximate dynamic programming (ADP) algorithm to the problem of job releasing and sequencing of a benchmark reentrant manufacturing line (RML). The ADP approach is based on the $SARSA(\lambda)$ algorithm with linear approximation structures that are tuned through a gradient-descent approach. The optimization is performed according to a discounted cost criterion that seeks both the minimization of inventory costs and the maximization of throughput. Simulation experiments are performed by using different approximation architectures to compare the performance of optimal strategies against policies obtained with ADP. Results from these experiments showed a statistical match in performance between the optimal and the approximated policies obtained through ADP. Such results also suggest that the applicability of the ADP algorithm presented in this paper may be a promising approach for larger RML systems.**

I. INTRODUCTION

It is well known that the control of queueing networks is a particularly difficult problem due to large state and action spaces commonly associated with these systems. Exact optimal control solutions for these systems are thus generally intractable [1]. Reentrant lines [2] are clear examples of queueing networks with these types of difficulties for control.

Reentrant lines are systems in which jobs can return to previous processing steps in the production flow. Such models are commonly utilized to represent complex production systems as those found in the semiconductor manufacturing industry. From this point forward we refer to these systems as reentrant manufacturing lines (RML). The fabrication of semiconductor devices is a clear example of a RML because such devices are built in silicon wafers through a repetitive manufacturing process; that is, jobs return several times to particular workstations before completing the fabrication process.

The control problem of RML in semiconductor manufacturing systems (SMS), also known as *Shop Floor Control* (SFC) [3], has received significant attention from researchers during the last decades, e.g., see [4], [2], [5], [6]. The research in this area has been motivated by both the economic and technological impact of semiconductor devices in many human activities, and by the challenging nature of the control of these systems. In general, SFC can be categorized in job sequencing [4], [2], [7], [6] and job releasing (input regulation) problems [8], [9]. While in the former problem the control is focused on deciding which job is processed next when various jobs are competing for service in a workstation, in the latter problem the control decides when to release new jobs into the system.

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

The difficulties on dimensionality of the state and control spaces associated with the control of RML make these systems potential candidates to apply optimization strategies based on approximate dynamic programming (ADP) [10], [11]. This idea is reinforced by the fact that in SMS there is an extensive utilization and maintenance of sophisticated simulation models [12] to assess overall performance. Therefore, ADP approaches based on the utilization of simulation models (i.e., simulation-based optimization) may provide important benefits in the control of such systems.

The objective of this paper is then to present the application of an ADP approach to the problem of controlling both releasing and sequencing of jobs in a RML system under a discounted-cost (DC) optimization criterion [13], [14]. The DC criterion is consider because it facilitates the analysis of the optimal control (e.g., see [15], [16], [17], [18]), and it could also result useful for improving performance in the short-term for SMS. The relevance of the optimization of short-term production performance in SMS is driven by the current dynamics in this industry where new products have short life-cycles [19], [20] and require a rapid ramp-up [21] in the manufacturing process.

The model utilized in this paper was originally presented in [17], and corresponds to an adapted version of the benchmark RML system with two workstations and three buffers given in [6], [15], [16], [22], [23]. The adapted version incorporates an additional buffer and server for modeling of the job releasing control. It is important to mention that the results on the optimal policy presented in [15], [17], [18] are incorrect and a corrigendum for these results is provided in [24]. Thus, the results given in [24] and numerical solutions of the optimal policy, obtained through the modified policy iteration algorithm [14], [25], are utilized in this paper to provide a baseline to compare the performance of policies obtained through the proposed ADP approach. Such comparisons were performed through a series of simulation experiments that showed statistical match in performance between optimal strategies and policies obtained with ADP. In addition, this paper continues the line of research presented in [15] where an ADP approach based on a lookup table version of Q-learning [10], [11], [26], [27] was utilized to approximate the optimal job sequencing policy of the benchmark RML system given in [6], [15], [16], [22], [23].

The remainder of this paper is organized as follows. Section II provides a brief overview of the literature on the application of ADP for control of RML systems. In section III, we present both the benchmark RML model with job releasing and sequencing control, and the underlying dynamic programming formulation for the control problem. The ADP algorithm and simulation results are presented in section IV. Conclusions are provided in section VI.

II. BRIEF REVIEW OF LITERATURE ON ADP FOR CONTROL OF RML SYSTEMS

Research on the application of ADP in the control of RML systems has been the subject of research in e.g., [15], [16], [28], [29]. In [28], the problem of job sequencing in a closed reentrant line with a fixed number of circulating jobs is presented. By using a simulation model, a learning approach based on $TD(\lambda)$ and linear approximation architectures [30] was utilized to obtain policies that maximize the throughput of a RML system with two stations, four buffers, and exponentially distributed processing times. Simulation results demonstrated that the ADP approach produced policies with good performance compared to policies based on heuristics and common dispatching rules; however, no optimal policies were provided to compare performance of the ADP approach.

Contrary to the work in [28], in [29] open reentrant lines are considered. This body of research presents the evaluation of different linear architectures in the approximation of the relative value function [13], [14] for an average cost (AC) optimization problem. This work considers simple RML models with finite capacity buffers and exponential processing times. The optimization seeks the maximization of throughput in the system while maintaining logical correctness in the process flow, i.e., deadlock-free operation [29]. The approximation of the optimal relative value functions is obtained by using a linear programming formulation and linear approximation architectures. This approach did not utilize a simulation-based strategy and it is subject to previous knowledge of the transition probabilities of the system.

In contrast to the work in [29], in [15], [16] a Q-learning and simulation-based approach is utilized to obtain near optimal job sequencing policies for the benchmark (open network) RML model given in [6], [22], [23] with exponential processing times and Poisson arrivals. In this case no previous knowledge of the transition probabilities of the system were required and the optimization problem considered the minimization of a discounted-cost criterion. Results from simulation experiments showed that a gradual approximation to the optimal policy is obtained as well as a statistical match in performance between the optimal and approximated policies. However, given that in [15] a lookup table version of Q-learning was utilized, extensive computational work was required, i.e., memory and processing. To overcome this difficulty, in [16] a state-aggregation approach is presented to obtain a compact state space representation that provide good performance in the case of the benchmark RML. Still, the proposed state aggregation approach may result non practical for larger RML systems.

The work presented in this paper contributes to the research in this area by extending the work in [15], [16] and by considering a more challenging problem on which the control space now includes both job sequencing and releasing. Given the computational difficulties found in [15], here we present the application of an ADP approach based on the $SARSA(\lambda)$ algorithm [10], [11] with parametric and linear approximations of the optimal Q-factors. In this approach, the parameters of the approximation structures are tuned through the use of a gradient-descent algorithm and temporal difference learning [10], [11].

III. BENCHMARK RML MODEL WITH JOB SEQUENCING AND RELEASING CONTROL & DYNAMIC PROGRAMMING FORMULATION

In this section we present an overview of both the model of the RML system utilized in this paper and the dynamic programming formulation of the control problem. Additional details can be found in [15], [16], [17].

A. *Benchmark RML with job sequencing and releasing control*

Figure 1 illustrates the model utilized in this paper. As mentioned earlier, such system was originally presented in [17] and represents an adapted version of the benchmark RML model given in e.g., [6], [15], [16], [22], [23].

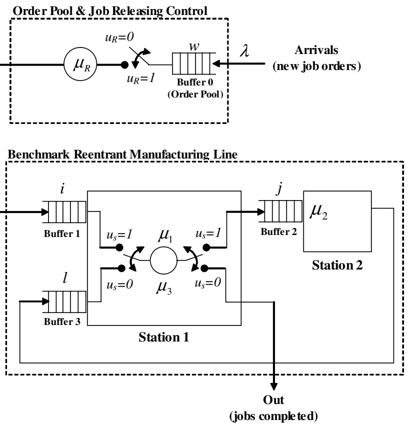


Fig. 1. Benchmark Reentrant Manufacturing Line with job releasing and sequencing control [17].

The system in Figure 1 corresponds to a Semi-Markov Decision Process (SMDP) with a continuous-time Markov chain. The state of the system is given by the tuple $s(t) := (w(t), i(t), j(t), l(t))$ corresponding to the buffer levels at time *t*, with $s(t) \in S$, where $S := \{(w, i, j, l) | \ \xi \leq L_\xi < +\infty$, with $\xi, L_\xi \in \mathbb{Z}^*$, and $\xi = w, i, j, l\}$ is the state space and $\mathbb{Z}^* := \mathbb{Z}^+ \cup \{0\}$. Thus, the dimension of S is determined by the buffer finite capacities L_ξ, with $\xi = w, i, j, l$, respectively. If a buffer reaches its maximum capacity L_ξ, then a blocking mechanism is activated and no jobs are allowed to be received in that buffer.

The production sequence of the system in Figure 1 is as follows: new order arrival→Buffer 0 (order pool), job releasing station→Buffer 1, Station 1→Buffer 2, Station 2→Buffer 3, Station 1→Out (job completed). The processing times at each station are exponentially distributed with means $\frac{1}{\mu_R}$, $\frac{1}{\mu_1}$, $\frac{1}{\mu_2}$, and $\frac{1}{\mu_3}$, respectively. Moreover, jobs waiting for service in buffers 1 and 3 are served at rates μ_1 and μ_3, respectively.

For this system, control decisions deal with both job releasing and sequencing into and inside the benchmark RML, respectively. In the former task a new job is released into the RML when $u_R = 1$, then an order is taken from the pool and it is converted into an effective job that is sent to the RML. If $u_R = 0$, then no orders are executed and no jobs are released. In addition, we assume that there is an infinite amount of raw material to cope with the corresponding demand. In the latter task, jobs waiting in buffers 1 and 3 are chosen to be served in Station 1 by selecting $u_s = 1$ and $u_s = 0$, respectively. Therefore, the control of the system is defined as a vector $\mathbf{u} := [u_R \;\; u_s]$ with $\mathbf{u} \in \mathbf{U}$, $\mathbf{U} := \mathcal{U}_R(s) \times \mathcal{U}_s(s)$, $u_R \in \mathcal{U}_R(s) \subseteq U_R$, and $u_s \in \mathcal{U}_s(s) \subseteq U_s$, where $U_R := \{0, 1\}$, $U_s := \{0, 1\}$, and $\mathcal{U}_R(s)$, $\mathcal{U}_s(s)$ are constraints for the control actions u_R and u_s, respectively, given $s \in S$. In particular, $\mathcal{U}_R(w, i, j, l) := \{0\}$ if $w = 0$ or $i = L_i$; i.e., the control u_R is constrained by the capacity in buffer 1 as well as the availability of new orders in buffer 0. In addition, we consider a non-idling policy as a constraint in the job sequencing control. That is, given $(w, i, j, l) \in S$, then $\mathcal{U}_s(w, i, j, l) = \{1\} \; \forall \; (w \geq 0, \; i > 0, \; j \geq 0, \; l = 0)$, and $\mathcal{U}_s(w, i, j, l) = \{0\} \; \forall \; (w \geq 0, \; i = 0, \; j \geq 0, \; l \geq 0)$.

B. Dynamic programming formulation

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

Definition 1: Given a discount factor $\beta > 0$, with $\beta \in \mathbb{R}$, then

$$J_\beta^\pi(s_0) := \lim_{N\to\infty} E_\pi \left\{ \int_0^{t_N} e^{-\beta t} g\left(s(t), \mathbf{u}(t)\right) dt \,\middle|\, s(0) = s_0 \right\}, \tag{1}$$

is the β-discounted cost under policy $\pi \in \Pi_{ad}$, where Π_{ad} is the set of admissible policies, t_N is the time for the N-th state transition, $g(s(t), \mathbf{u}(t))$ is the continuous-time one-stage cost function, and s_0 is the initial state, $s(t) \in S$, $\mathbf{u}(t) \in \mathbf{U}$. In addition, the optimal β-discounted cost is defined as $J_\beta^*(s_0) := \min_\pi J_\beta^\pi(s_0)$. Moreover, if $J_\beta^{\pi^*}(s_0) = J_\beta^*(s_0)$, then $\pi^* \in \Pi_{ad}$ is said to be an optimal policy.

Because the RML can be represented by a continuous-time Markov chain with exponentially distributed processing times, then a *uniformization* [14], [31] procedure is performed to obtain a discrete-time and statistically equivalent model. The corresponding discrete-time model is defined as follows [14]:

Definition 2: Given a uniform version of a SMDP under the discounted cost criteria (1), then

$$J_\alpha^\pi(s) := \lim_{N\to\infty} E_\pi \left\{ \sum_{k=0}^{N} \alpha^k \tilde{g}(s_k, \mathbf{u}_k) \,\middle|\, s_0 = s \right\} \tag{2}$$

is the α-discounted cost under policy $\pi \in \Pi_{ad}$, where $\alpha := \frac{\nu}{\beta+\nu}$, and

$$\tilde{g}\left(s, \mathbf{u}\right) := \frac{g\left(s, \mathbf{u}\right)}{\beta + \nu} + \hat{g}(s, \mathbf{u}), \tag{3}$$

are, namely, the discount factor and the one-stage cost function for the discrete-time model. In (3), the cost function $\hat{g}(s, \mathbf{u})$ is utilized to model situations where a cost (or profit), which is independent of the length of the state transition interval, is imposed at the moment of applying control $\mathbf{u}$ at state s [14]. In addition, $s_k \in S$ and $\mathbf{u}_k \in \mathbf{U}$ are the state and control at the k-th state transition, ν is the uniform transition rate, with $\nu \geq \nu_s(\mathbf{u})$ for all $s \in S$, $\mathbf{u} \in \mathbf{U}$, and $\nu_s(\mathbf{u})$ is the rate of transition [14] associated to state s and control $\mathbf{u}$. For the RML system described in section III, ν is defined as follows:

$$\nu := \lambda + \mu_R + \mu_1 + \mu_2 + \mu_3. \tag{4}$$

The optimal α-discounted cost $J_\alpha^*(s)$, with $s \in S$, is defined as $J_\alpha^*(s) := \min_\pi J_\alpha^\pi(s)$, and from Definition 1 and the uniformization procedure, we have that $J_\alpha^*(s) = J_\beta^*(s)$.

Figure 2 depicts the state transitions diagram for the uniformized version of the continuous-time Markov chain of the benchmark RML with job releasing and sequencing control, and where R, A, B_1, B_2 and B_3 are mappings from S to S [32], as follows:

- $Rs = (w - 1, i + 1, j, l)$,
- $As = (\min\{w + 1, L_w\}, i, j, l)$,
- $B_1 s = (w, (i - \mathbb{1}_j)^+, \min\{j + \mathbb{1}_{i>0}, L_j\}, l)$,
- $B_2 s = (w, i, (j - \mathbb{1}_l)^+, \min\{l + \mathbb{1}_{j>0}, L_l\})$, and
- $B_3 s = (w, i, j, (l - 1)^+)$,

where $(\cdot)^+ := \max(\cdot, 0)$, $\mathbb{1}_\xi := \mathbb{1}(\xi < L_\xi)$, $\mathbb{1}_{\xi>0} := \mathbb{1}(\xi > 0)$, and $\mathbb{1}(\cdot)$ is the indicator function.

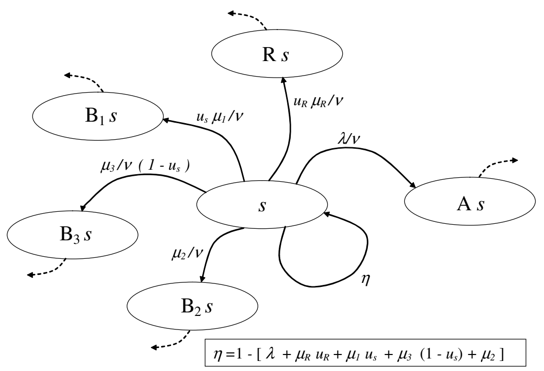

R s
B1 s
$u_s \mu_1/\nu$
$u_R \mu_R/\nu$
λ/ν
μ_3/ν (1 - u_s)
s
A s
B3 s
μ_2/ν
η
B2 s
η =1 - [λ + μ_R u_R + μ_1 u_s + μ_3 (1 - u_s) + μ_2]

Fig. 2. State transitions diagram for uniformized version of the continuous-time Markov chain associated to the benchmark RML with job releasing and sequencing control [17].

Thus, given $s \in S$ and $\mathbf{u} \in \mathbf{U}$ and the state transition probabilities depicted in Figure 2, the Bellman's optimality equation for the job releasing and sequencing problem of the benchmark RML is as follows:

$$\begin{array}{l} J_\alpha^*(s) = \frac{1}{\beta+\nu} \left[\, g(s) - p \cdot \phi(d) + \lambda J_\alpha^*(As) + \mu_1 J_\alpha^*(s) \right. \\ \quad + \mu_2 J_\alpha^*(B_2 s) + \mu_3 J_\alpha^*(B_3 s) + \mu_R J_\alpha^*(s) \\ \quad \left. + \min_{\mathbf{u} \in \mathbf{U}} \{ \mu_R \cdot u_R \cdot \Delta_R(s) + u_s \cdot \Delta_s^d(s) \} \right], \\ \quad \forall\, s \in S, \;\; d \in \{0,1\}, \end{array} \tag{5}$$

where,

$$\Delta_R(s) := J_\alpha^*(Rs) - J_\alpha^*(s), \tag{6}$$

$$\Delta_s^d(s) := \Delta_s(s) + p \cdot \phi(d), \tag{7}$$

and

$$\Delta_s(s) := \mu_1 [J_\alpha^*(B_1 s) - J_\alpha^*(s)] - \mu_3 [J_\alpha^*(B_3 s) - J_\alpha^*(s)], \tag{8}$$

where p is a profit per job completed, $g(s)$ is a one-stage inventory cost function and $\phi(d)$ is defined as follows:

$$\phi(d) := \mu_3\, d + (1-d)(\beta+\nu), \; d \in \{0,1\}. \tag{9}$$

Thus, $d = 1$ corresponds to the case where profits are discounted during state transition intervals, and $d = 0$ corresponds to the situation when profits are not discounted during such intervals. Notice that $\phi(0) \geq \phi(1)$. In addition, $g(s)$ is assumed to be nonnegative and monotonically nondecreasing w.r.t componentwise partial order (see [17] for details).

From (5)-(8), the optimality conditions are as follows: consider $s \in S$ s.t. the control constraints on $\mathbf{u}$ are not applied, then it is optimal to release a new job iff $\Delta_R(s) \leq 0$, and it is optimal to serve a job in buffer l iff $\Delta_s(s) \geq 0$.

IV. ADP APPROACH: PARAMETRIC APPROXIMATIONS OF OPTIMAL Q-FACTORS

This section presents details of the ADP approach utilized to approximate the optimal Q-factors as well as the results from simulation experiments. The objectives of these experiments were to obtain near optimal policies by using the ADP algorithm, and to compare the performance (i.e., discounted-cost) of the approximations against the optimal strategies. In the experiments we considered either linear or quadratic inventory cost functions $g(s)$, i.e., $g(s) = c_w w + c_i i + c_j j + c_l l$, and $g(s) = c_w w^2 + c_i i^2 + c_j j^2 + c_l l^2$, respectively, with $c_w, c_i, c_j, c_l \in \mathbb{R}^+$.

A. ADP algorithm: gradient-descent SARSA(λ)

The ADP approach utilized to approximate the optimal Q-factors and subsequently the optimal policy is that based on the *SARSA(λ)* algorithm (State-Action-Reward-State-Action) [10], [11] with parametric approximations. The SARSA(λ) algorithm was selected because, among other similar methods such as Q(λ), Watkin's Q(λ), and Peng's Q(λ) [10], it has the advantage of being computationally least expensive and of providing faster convergence [10], [33]. For a detailed comparison of different ADP algorithms utilized to approximate the optimal Q-factors see [33]. Thus, we consider linear approximation architectures as follows:

$$Q^*(s,\mathbf{u}) \approx \widehat{Q}_{\mathbf{u}}(s, \mathbf{r_u}) := \sum_{m=1}^{M} \psi_{m,\mathbf{u}}(s) \cdot r_{m,\mathbf{u}}, \tag{10}$$

where $\widehat{Q}_{\mathbf{u}}(s, \mathbf{r_u})$ is the parametric approximation of the optimal Q-factor given $s \in S$ and $\mathbf{u} \in \mathbf{U}$, where $J_\alpha^*(s) := \min_{\mathbf{u} \in \mathbf{U}} Q^*(s,\mathbf{u})$. Thus, there are as much $\widehat{Q}_{\mathbf{u}}(s, \mathbf{r_u})$ functions as combinations of the control components of $\mathbf{u}$, i.e., u_s and u_R. In addition, $\mathbf{r_u} := [r_{1,\mathbf{u}}, ..., r_{M,\mathbf{u}}]$ is a M-dimensional vector of parameters, with $r_{m,\mathbf{u}} \in \mathbb{R}$, and $\psi_{m,\mathbf{u}}(s)$ is a feature or basis function [10], [11], [30], [34], [35] s.t. $\psi_{m,\mathbf{u}} : S \to \mathbb{R}$, and $m = 1, ..., M$; with M as the number of features in the approximation architecture.

Each parameter vector $\mathbf{r_u}$ is adjusted through a gradient-descent approach that seeks to minimize the mean-squared error between the estimated and the optimal value of the Q-factors. Moreover, an analogous approach to that of *temporal-differences* learning algorithms [10], [11], [36] was utilized to guide the tunning of the parameters $r_{m,\mathbf{u}}$. Such temporal differences are defined as follows:

$$\begin{array}{l} d_{\mathbf{u},t}(s, \mathbf{r_u}) := \widetilde{g}(s_t, \mathbf{u}, s_{t+1}) \\ \qquad\qquad + \alpha \widehat{Q}_{\mathbf{u}'}(s_{t+1}, \mathbf{r}_{\mathbf{u}'}) - \widehat{Q}_{\mathbf{u}}(s_t, \mathbf{r_u}), \end{array} \tag{11}$$

where $\mathbf{u}' = \arg\min_{\mathbf{u} \in \mathbf{U}(s_{t+1})} \widehat{Q}_{\mathbf{u}}(s_{t+1}, \mathbf{r_u})$, $\widetilde{g}(\cdot)$ is the one-stage cost-function, $\alpha \in (0,1)$ is the discount factor, s_t and s_{t+1} are the current and next state, respectively; and t is the index for state-transitions. Thus, each vector of parameters is updated in the following form:

$$\mathbf{r}_{\mathbf{u},t+1} := \mathbf{r}_{\mathbf{u},t} + \gamma_t(\mathbf{u}) \cdot d_{\mathbf{u},t}(s_t, \mathbf{r}_{\mathbf{u},t}) \cdot \mathbf{z}_{\mathbf{u},t}, \tag{12}$$

where $\gamma_t(\mathbf{u})$ is the step size given the control $\mathbf{u} \in \mathbf{U}$ and s.t. $\sum_{t=0}^{\infty} \gamma_t(\mathbf{u}) = \infty$ and $\sum_{t=0}^{\infty} \gamma_t^2(\mathbf{u}) < \infty$. In addition, $\mathbf{z}_{\mathbf{u},t}$ is the vector of eligibility traces [10], [11] given the control $\mathbf{u}$ which is updated as follows:

$$\mathbf{z}_{\mathbf{u},t+1} := \alpha \cdot \lambda_{ADP} \cdot \mathbf{z}_{\mathbf{u},t} + \overrightarrow{\psi}_{\mathbf{u}}(s_{t+1}), \tag{13}$$

with $\overrightarrow{\psi}_{\mathbf{u}}(s) := [\psi_{1,\mathbf{u}}(s) \;...\; \psi_{M,\mathbf{u}}(s)]$, and $\lambda_{ADP} \in [0,1]$.

For each update of the parameters an ϵ-greedy policy [10], [11] approach is utilized. Therefore, exploration on the control space is allowed by selecting a random control action with a small probability ϵ.

B. Approximation architectures

Three different sets of basis functions were utilized in the simulation experiments. Each set provided a different approximation architecture, namely A1, A2, A3. Thus, the vectors of basis functions were defined as follows:

- $\overrightarrow{\psi}_{\mathbf{u}}^{A1}(s) := [1]$,
- $\overrightarrow{\psi}_{\mathbf{u}}^{A2}(s) := [w \;\; i \;\; j \;\; l \;\; 1]$,
- $\overrightarrow{\psi}_{\mathbf{u}}^{A3}(s) := [w^2 \;\; i^2 \;\; j^2 \;\; l^2 \;\; w \;\; i \;\; j \;\; l \;\; 1]$,

and the resulting approximation architectures can be written in the following form:

$A1: \ \widehat{Q}_{\mathbf{u}}(s,\mathbf{r_u}) = r_{0,\mathbf{u}},$
$A2: \ \widehat{Q}_{\mathbf{u}}(s,\mathbf{r_u}) = r_{0,\mathbf{u}}w + r_{1,\mathbf{u}}i + r_{2,\mathbf{u}}j + r_{3,\mathbf{u}}l + r_{4,\mathbf{u}},$
$A3: \ \widehat{Q}_{\mathbf{u}}(s,\mathbf{r_u}) = r_{0,\mathbf{u}}w^2 + r_{1,\mathbf{u}}i^2 + r_{2,\mathbf{u}}j^2 + r_{3,\mathbf{u}}l^2 + r_{4,\mathbf{u}}w + r_{5,\mathbf{u}}i + r_{6,\mathbf{u}}j + r_{7,\mathbf{u}}l + r_{8,\mathbf{u}},$

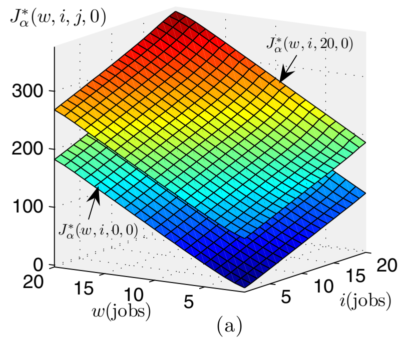
where $s=(w,i,j,l)\in S$, $\mathbf{u}=[u_R \ \ u_s]\in\mathbf{U}$.

If the approximation functions are defined as follows:

$$\widehat{Q}_{\mathbf{u}}(s,\mathbf{u}) := \widehat{Q}_{u_R,u_s}(s,\mathbf{r}_{u_R,u_s}), \tag{14}$$

then we obtain four Q-factors approximation functions, i.e., $\widehat{Q}_{0,0}(s,\mathbf{r}_{0,0})$, $\widehat{Q}_{0,1}(s,\mathbf{r}_{0,1})$, $\widehat{Q}_{1,0}(s,\mathbf{r}_{1,0})$, and $\widehat{Q}_{1,1}(s,\mathbf{r}_{1,1})$. Likewise, there are four vectors of parameters that need to be tunned: $\mathbf{r}_{0,0}$, $\mathbf{r}_{0,1}$, $\mathbf{r}_{1,0}$, and $\mathbf{r}_{1,1}$.

As can be noticed, the list of architectures have an increasing number of degrees of freedom (i.e., number of features or basis functions) to approximate the optimal Q-factors from A1 to A3. We selected these architectures because the optimal cost $J^*_\alpha(s)$ seems to follow closely the structure of the one-stage cost function $\widetilde{g}(s,\mathbf{u})$. For instance, Figures 3(a)-(b) and Figures 4(a)-(b) show examples of the form of the optimal cost $J^*_\alpha(s)$ (for $j\in\{0,\ 20\}$ and $l=0$), computed through value iteration for linear and quadratic one-stage cost functions, and the function $g(s)$ when linear and quadratic (for $j\in\{0,\ 20\}$ and $l=0$), respectively. As illustrated in the figures, there is an important similarity between the optimal cost $J^*_\alpha(s)$ and $g(s)$. Thus, we assumed that linear and quadratic architectures may provide good approximations to the optimal Q-factors for the given inventory cost functions.



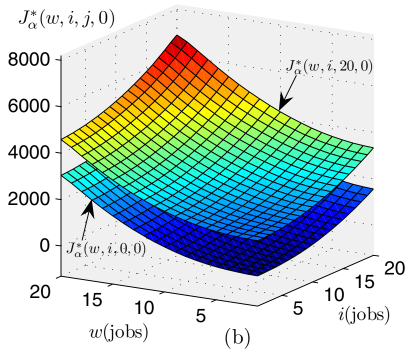
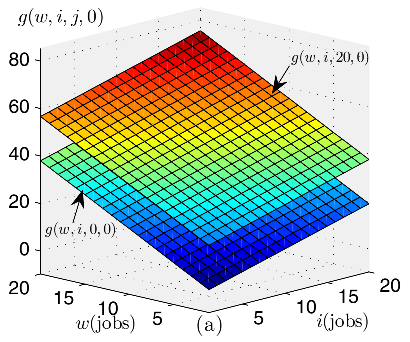


Fig. 3. Examples of the optimal cost $J^*_\alpha(w,i,j,l)$ with $j\in\{0,\ 20\}$, $l=0$ and computed through value iteration, when: (a) $g(s)$ linear, (b) $g(s)$ quadratic.



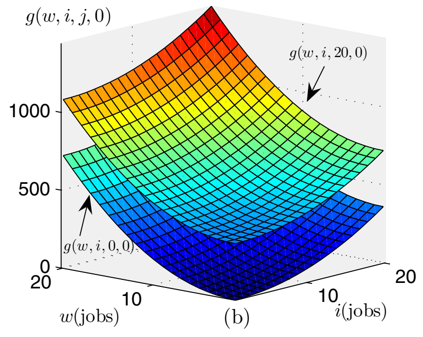


Fig. 4. Linear and quadratic one-stage cost functions $g(w,i,j,l)$ with $j\in\{0,\ 20\}$, $l=0$: (a) linear, (b) quadratic.

C. Simulation conditions

Two steps were followed for the simulation experiments. While in the first step simulations were performed to tune the approximations to the optimal Q-factors and policies, in the second step such approximations were evaluated through simulation and compared in performance against the optimal strategies.

The conditions for the experiments were as follows:

- Parameters of the RML system: $\mu_R = 0.4492$, $\mu_1 = \mu_3 = 0.3492$, $\mu_2 = 0.1587$, $\lambda = 0.1430$, $L_w = L_i = L_j = L_l = 20$ jobs $\Rightarrow$ 160000 states.
- Initial state of the system: $s_0 = (1,0,0,0)$.
- Discount factor: $\beta = 0.2 \Rightarrow \alpha \approx 0.878$.
- Profits per job completed: Two cases, $p=0$ and $p=25$ with discounted profits between state transition intervals, i.e., $d=1$.
- For each approximation architecture, A1-A3, a total of 100 replications with a length of 2000 time units were utilized to tune the vectors $\mathbf{r_u}$. Through trial and error it was determined that for $\beta=0.2$ the discounted-cost converges before completing 2000 time units of simulation. Both the number of replications and simulation length

were sufficient to obtain appropriate convergence of the parameters in the approximation structures.

- All vectors of parameters were initialized with all components equal to zero, i.e., $\mathbf{r}_{\mathbf{u},0} = [0 \ ... \ 0]$ for all $\mathbf{u}$.
- Step-size in the ADP algorithm: $\gamma_t(\mathbf{u}) := \frac{p_\gamma}{\mathrm{v}_t(\mathbf{u})}$, where $\mathbf{v}_t(\mathbf{u})$ is the number of visits to the control $\mathbf{u}$ at the t-th state transition, and $p_\gamma \in \mathbb{R}^+$ is a small number.
- Different sets of parameters $\mathbf{r}_\mathbf{u}$ were obtained by varying the parameters of the algorithm as follows:
  - $\lambda_{ADP} \in \{0.1, 0.4, 0.7, 0.9\}$,
  - $\epsilon \in \{0.0001, 0.0010, 0.0100, 0.1000\}$,
  - $p_\gamma \in \{0.01, 0.001, 0.0001\}$.
- Evaluation of the approximated optimal policies were performed by running between 100 and 250 replications of 2000 time units each.
- The simulation software ARENA [37], and its application "Process Analyzer," were utilized for both encoding the ADP algorithm and performing the experiments. Among other simulation software, ARENA was selected not only because of its availability with no cost for research purposes, but because it offers an easy to use interface that facilitates the construction of the simulation model and implementation of ADP algorithms through its Visual Basic module. Moreover, the "Process Analyzer" tool from ARENA facilitates performing simulations under different scenarios (e.g., various sets of parameters) which is an important part of the experimentation with ADP algorithms. Nevertheless, any other simulation software with some kind of low-level programming module should provide similar results.

D. Simulation results: linear inventory cost function

First, we present the results when the inventory cost function is linear as follows: $g(s) = 2w + i + j + l$. From the results in [17], [24] and numerical solutions by using the modified policy iteration (MPI) algorithm [14], [25], the optimal policy is both to release new jobs whenever there is at least one job available in buffer 0, and to always serve buffer 3 for either $p = 0$ or $p = 25$ with $d = 1$. Table I shows the value of the optimal discounted cost computed by performing multiple simulation replications when the optimal policy was included in the simulation model of the benchmark RML. The corresponding 95% confidence intervals are also provided in Table I.

TABLE I
COMPUTED OPTIMAL DISCOUNTED COST FOR LINEAR INVENTORY COSTS

(p, d)	$\widetilde{J}^*_\alpha(s_0)$
$(0, -)$	9.45 ± 0.62
$(25, 1)$	6.65 ± 0.81

p: profits per job completed, d: selector between profits discounted during state transitions intervals (d=1 discounted, d=0 not discounted).

Among all the approximations of the optimal policy through ADP and according to the simulation conditions, the best performances obtained for each architecture are indicated in Table II for the cases when $p = 0$ and $p = 25, \ d = 1$. In Table II, $J^{\pi_{A1}}_\alpha$, $J^{\pi_{A2}}_\alpha$, and $J^{\pi_{A3}}_\alpha$ are the discounted costs obtained by the approximations to the optimal policy with ADP and for the architectures $A1$, $A2$, and $A3$, respectively.

TABLE II
PERFORMANCE FOR APPROXIMATIONS TO THE OPTIMAL POLICY OBTAINED WITH ADP: LINEAR INVENTORY COSTS

(p, d)	$J^{\pi_{A1}}_\alpha(s_0)$	$J^{\pi_{A2}}_\alpha(s_0)$	$J^{\pi_{A3}}_\alpha(s_0)$
$(0, -)$	10.25±0.62	10.09±0.64	10.15±0.65
$(25, 1)$	7.08±0.89	6.87±0.86	6.77±0.87

π_{A1}, π_{A2}, π_{A3}: approximations to the optimal policies obtained with ADP when architectures A1, A2, and A3 were utilized, respectively.

As can be noted on Tables I and II, the approximations with ADP obtained a statistical match in performance with the optimal policy in all the cases. Results show that when $p = 0$, the architecture $A2$ seems to provide the best performance. Similarly, for the case when $p = 25, \ d = 1$ the best performance is given by the architecture $A3$. Under both conditions for p, the best set of parameters $r_{m,\mathbf{u}}$ were obtained when $\lambda_{ADP} = 0.7$. This coincides with the experimental fact indicated in the literature [10] that such value of λ_{ADP} yields in general a better performance in temporal difference learning algorithms.

In addition, Figures 5 and 6 show the evolution of both the discounted cost and the parameters for architecture $A2$ through simulation replications and state transitions, respectively. Figure 5 shows how the algorithm starts with higher values for the discount cost that later become closer to the optimal value as the parameters are properly tuned to minimize the estimation error. In Figure 6, the evolution of the parameters indicates that after 10000 state transitions the parameters reach near convergence values.

E. Simulation results: quadratic inventory cost function

Now, consider the case where the inventory cost function is quadratic as follows: $g(s) = w^2 + i^2 + j^2 + l^2$. From the results given in [17], [24], and numerical solutions through the MPI algorithm, the optimal input regulation policy in this case corresponds to releasing a new job if $w \geq i + 1$, and the optimal job sequencing policy can be expressed as serving buffer 3 if $j + l + \frac{1}{2} + 1.6493 \cdot p \geq i$. Thus, Table III lists the optimal discounted cost computed through multiple simulation replications, given p and d.

Table IV presents the estimated discounted cost obtained by the approximations of the optimal policy with ADP and for architectures $A1$, $A2$, and $A3$. As in the case of linear inventory costs, a statistical match in performance is obtained between the optimal policy and the approximations obtained with ADP. It should be noted, however, that variability of the estimations is higher compared with the case of linear costs given that the one-stage cost function is quadratic. Also notice that the best performance is consistently obtained by

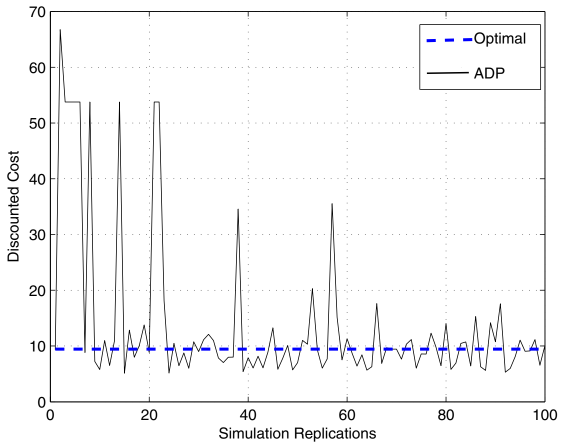


Fig. 5. Typical evolution of the approximation to the optimal cost by ADP with parametric approximations when a linear inventory cost function is utilized and $p = 0$.

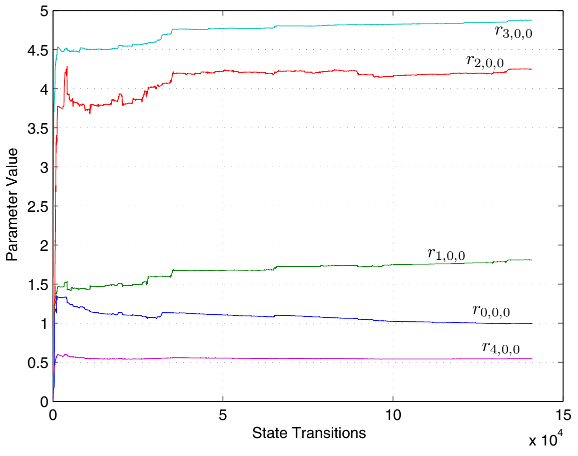


Fig. 6. Evolution of parameters of vector $\mathbf{r}_{0,0}$ in architecture $A2$ when a linear inventory cost function is utilized, $p = 0$, $\lambda_{ADP} = 0.7$, $\epsilon = 0.1$, and $p_\gamma = 0.01$.

architecture $A3$. This suggests that the selection of features utilized in the approximation structures is closely related to the structure of the one-stage cost function. A similar situation can be observed in the case of linear inventory costs, where the simplest architecture, $A1$, provided a policy with a performance close to those associated the more complex architectures, i.e., $A2$ and $A3$. Notice that in the case of linear inventory costs the optimal policy is static; therefore, it is expected that the policy obtained with architecture $A1$ will provide good performance. In addition, and as in the case of linear inventory costs, when $p = 25$, $d = 1$, the best set of parameters is obtained for $\lambda_{ADP} = 0.7$.

An important observation of the approximations obtained with ADP is the reduced computational effort when compared to numerical solutions obtained with the MPI. While the solution time with the MPI exponentially increases with the buffer size (i.e., utilizes a complete enumeration of the state space), the ADP approach seems to be computationally more convenient given that it exploits the search for near-optimal solutions over a subset of the state space, and regardless of the capacity in the buffers of the system. As a result, the solution time for near-optimal policies is reduced. Similarly, approximation architectures with a high number of features may increase the necessary time to tune the parameters of the parametric architecture, and thus this may also increase the time required to achieve near optimal solutions.

TABLE III
COMPUTED OPTIMAL DISCOUNTED COST FOR QUADRATIC INVENTORY COSTS

(p,d)	$\widetilde{J}^*_\alpha(s_0)$
$(0,-)$	10.81 ± 0.77
$(25,1)$	7.43 ± 0.91

p: profits per job completed, d: selector between profits discounted during state transitions intervals (d=1 discounted, d=0 not discounted).

TABLE IV
PERFORMANCE FOR APPROXIMATIONS TO THE OPTIMAL POLICY OBTAINED WITH ADP: QUADRATIC INVENTORY COSTS

(p,d)	$J^{\pi A1}_\alpha(s_0)$	$J^{\pi A2}_\alpha(s_0)$	$J^{\pi A3}_\alpha(s_0)$
$(0,-)$	11.84±1.02	11.53±0.96	11.28±0.84
$(25,1)$	8.66±1.80	8.56±1.2	7.99±1.01

π_{A1}, π_{A2}, π_{A3}: approximations to the optimal policies obtained with ADP when architectures A1, A2, and A3 were utilized, respectively.

V. CONCLUSIONS

This paper presented the application of an ADP algorithm to the problem of job releasing and sequencing of a benchmark RML. The ADP approach is based on the $SARSA(\lambda)$ algorithm that utilizes temporal differences learning with a gradient-descent approach to tune the parameters in the approximation structures. Different architectures were utilized to obtain near optimal policies when linear and quadratic one-stage inventory cost functions were considered. Simulation results showed that a statistical match in performance is obtained between optimal strategies and policies obtained with ADP. Such results also suggest that the applicability of the ADP algorithm presented in this paper may be a promising approach for larger RML systems.

REFERENCES

[1] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of optimal queueing network control," *Mathematics of Operations Research*, vol. 24, no. 2, pp. 293–305, 1999.
[2] P. R. Kumar, "Re-entrant lines," *Queueing Systems: Theory and Applications*, vol. 13, pp. 87–110, 1993.
[3] R. Uzsoy, C. Lee, and L. A. 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.
[4] L. M. Wein, "Scheduling semiconductor wafer fabrication," *IEEE Transactions on Semiconductor Manufacturing*, vol. 1, pp. 115–130, 1988.

[5] C. H. Lu, D. Ramaswamy, and P. R. 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.

[6] S. Kumar and P. R. 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.

[7] P. R. Kumar, "Scheduling semiconductor manufacturing plants," *IEEE Control Systems Magazine*, vol. 39, no. 11, pp. 33–40, 1994.

[8] R. Uzsoy, C. Lee, and L. A. Martin-Vega, "A review of production planning and scheduling models in the semiconductor industry part ii: shop-foor control," *IIE Transactions*, vol. 24, pp. 45–55, 1994.

[9] J. W. Fowler, G. L. Hogg, and S. J. Mason, "Workload control in the semiconductor industry," *Production Planning & Control*, vol. 13, no. 7, pp. 568–578, 2002.

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

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

[12] J. A. Ramírez-Hernández, H. Li, E. Fernandez, C. R. McLean, and S. Leong, "A framework for standard modular simulation in semiconductor wafer fabrication systems," in *Proceedings of the 2005 Winter Simulation Conference*, Orlando, FL, December 2005, pp. 2162–2171.

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

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

[15] J. A. Ramírez-Hernández and E. Fernandez, "A case study in scheduling reentrant manufacturing lines: Optimal and simulation-based approaches," in *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005*, Seville, Spain, December 12-15 2005, pp. 2158–2163.

[16] ——, "Optimal job sequencing control in a benchmark reentrant manufacturing line," January 2007, manuscript.

[17] ——, "Optimal job releasing and sequencing for a reentrant manufacturing line with finite capacity buffers," in *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, CA, December 13-15, 2006, pp. 6654–6659.

[18] ——, "Optimal job sequencing in a benchmark reentrant line with finite capacity buffers," in *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems*, Kyoto, Japan, July 24-28, 2006, pp. 1214–1219.

[19] Semiconductor Industry Association (SIA). (2005) International Technology Road Map for Semiconductors (ITRS) 2005. [Online]. Available: http://public.itrs.net

[20] M. Venables, "Small is beautiful: small low volume semiconductor manufacturing plants," *IE Review*, pp. 26–27, March 2005.

[21] 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*, March 31^{st}-April 1^{st} 2003, pp. 111–117.

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

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

[24] J. A. Ramírez-Hernández and E. Fernandez. (2007) Corrigendum to optimal job releasing and sequencing for a reentrant manufacturing line with finite capacity buffers. [Online]. Available: http://www.ececs.uc.edu/~ramirejs/CorrigendumOptimalControlRML.pdf

[25] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY: John Wiley & Sons, Inc, 1994.

[26] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. thesis, Cambridge University, Cambridge, UK, 1989.

[27] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[28] C. Liu, H. Jin, Y. Tian, and H. Yu, "A reinforcement learning approach to re-entrant manufacturing system scheduling," in *Proceedings of ICII 2001 International Conferences on Info-tech and Info-net*, vol. 3, Beijing, China, October 29th to November 1st 2001, pp. 280–285.

[29] J.-Y. Choi and S. Reveliotis, "Relative value function approximation for the capacitated re-entrant line scheduling problem," *IEEE Transactions on Automation Science and Engineering*, vol. 2, no. 3, pp. 285–299, 2005.

[30] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Transactions on Automatic Control*, vol. 42, no. 5, pp. 674–690, 1997.

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

[32] J. Walrand, *An Introduction to Queueing Networks*. Englewood Cliffs, NJ: Prentice Hall, 1988.

[33] G. A. Rummery, "Problem solving with reinforcement learning," Ph.D. thesis, Cambridge University, 1995.

[34] B. Van Roy, "Learning and value function approximation in complex decision processes," Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, 1998.

[35] J. N. Tsitsiklis and B. Van Roy, "Feature-based methods for large scale dynamic programming," *Machine Learning*, vol. 22, pp. 59–94, 1996.

[36] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, pp. 9–44, 1988.

[37] W. D. Kelton, R. P. Sadowski, and D. T. Sturrock, *Simulation with Arena*, 4th ed. USA: McGraw-Hill, 2006.