Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, eds.

OPTIMIZATION MODEL SELECTION FOR SIMULATION-BASED APPROXIMATE DYNAMIC PROGRAMMING APPROACHES IN SEMICONDUCTOR MANUFACTURING OPERATIONS

Xiaoting Chen
Emmanuel Fernandez

School of Electronics and Computing Systems
University of Cincinnati
Cincinnati, OH 45221, U.S.A.

W. David Kelton

Department of Operations, Business Analytics
and Information Systems
University of Cincinnati
Cincinnati, OH 45221, U.S.A.

ABSTRACT

Guided by Little's law, decision and control models for operations in reentrant line manufacturing (RLM) systems are commonly set up to minimize the total work-in-process (WIP), which in turn indirectly minimizes cycle time (CT). By viewing the problem fundamentally differently, we re-formulate it as one that seeks to select the best cost function leading to optimal cycle times. We present the details and results of an extended simulation study, based on a benchmark problem, using a simulation-based approximate dynamic programming method, with a newly proposed extended actor-critic architecture. Our results support the idea that a Markov decision process modeling approach can be used as a flexible platform to explore different cost formulations, leading to a selection of an optimal cost and model to optimize cycle time directly.

1 INTRODUCTION

The control and decision problems arising in large scale manufacturing are very challenge tasks (Kumar 1994; Wein 1988). These processes involve large state and action spaces, thus making them analytically and computationally difficult. Manufacturing systems with feedback loops, i.e., reentrant line manufacturing (RLM) system, are of particular interest. Reentrant lines are manufacturing systems in which jobs will return to the early stations during their production. In particular, semiconductor manufacturing facilities (fabs) are usually modeled as RLMs because of the demand for repetitive layer processing by the same station of each silicon wafer (Kumar 1993).

Much research effort has been devoted to modeling and optimization of operations in semiconductor manufacturing systems, namely, the control of RLM systems. The control problems arising in RLM systems of this type are known as shop floor control (Uzsoy et al. 1994), which considers two major problems, job sequencing and scheduling, to improve overall performance of the system, e.g., cycle time (CT). The sequencing problem is of particular interest here, because the optimal solution is usually hindered by large state and action spaces. The existing literature with the sequencing problem is mainly based on heuristics and control theory, which are extensively studied by Lu and Kumar (1991), Kumar and Kumar (2001), Flores-Godoy et al. (1998), Tsakalis et al. (1997), and Vargas-Villamil et al. (2003).

Modeling and optimizing RLM systems with Markov decision process (MDP) (Puterman 1994) and its solution methods approximate dynamic programming (ADP) (Bertsekas 2007; Powell 2007) are reported by Ramirez-Hernandez and Fernandez (2007), Choi and Reveliotis (2005). Later on, due to its scalability for large systems, these methods were applied to more complex RLM systems (Ramirez-Hernandez and Fernandez 2009, 2010).

Due to the feedback and competitive nature of RLM systems, optimization for sequencing control with respect to cycle time is a highly complex process, and direct optimization in terms of the time metric is usually impractical. Existing methods are based upon an approximate transformation and mapping rule, the

978-1-4673-4782-2/12/$31.00 ©2012 IEEE

well-known Little's law (Little 1961). On the contrary, formulating the sequencing optimization problem as an MDP would require an explicit and precise functional preference, i.e., a one-stage cost function, over its state space in order to provide optimal results. Guided by Little's law, usually only the total number in queue is used, namely, the work-in-process (WIP), and thought to be proportional to cycle time, on average. However, cycle time is the true desired objective to be optimized, and for real applications, any departures from an ideal exponential queueing could lead to a non-optimal policy, in terms of cycle time. Even if an MDP formulation dictates, the derived policy should minimize the total number in queue. As shown later via simulation experiments, the choice of different cost functions, linear and others, leads to different policies with significant differences in average cycle time. Hence, we seek to select directly the cost function itself, which then leads to a derived policy that optimizes cycle time.

However, existing work with RLMs overlooks this issue and considers only a single functional form for the one-stage cost function, without considering other modeling and optimization possibilities with MDP methods. In this paper, we consider the sequencing optimization problem for reducing the cycle time of the open queue system (Whitt 1984), presented by the Intel Mini-Fab (Kempf 1994). Different from the existing work, we investigate the performance improvement and degradation with the Mini-Fab by exploring the use of different one-stage cost functions with a simulation-based ADP method, under a new proposed extended actor-critic architecture. We first discuss a departure from the reliance on Little's law to have some basic cost (function) realizations, followed by the detailed simulation results and performance comparison. Then, we show that these basic cost realizations can be easily extended to more general forms, and the proposed extended actor-critic architecture provides a flexible platform for learning and optimization with problems of this kind.

In Section 2, we briefly review Little's law and discuss two basic cost realizations used in an MDP formulation; we also discuss necessary changes to the conventional Markov decision process to have a new extended actor-critic architecture. In Section 3, we discuss the benchmark RLM model, the Intel Mini-Fab, followed by detailed optimization and simulation results in Sections 4 and 5, and conclusions are given in Section 6.

2 BACKGROUND

In this section, we'll first review and discuss multiple cost realizations with the MDP formulation of the control problem in an RLM system, in order to reach beyond a formulation depending exclusively on Little's law. Then, we go over conventional Markov decision process results, and introduce some modeling modifications, including new performance measurement and optimality with respect to multiple cost realizations.

2.1 Little's Law and Cost Realizations

Little's seminal work provides a solid foundation for understanding the relationship between cycle time and work-in-process in queueing systems. This relationship can be expressed through a simple equation $L = \lambda W$, where L is the expected number in the system, λ is the expected arrival rate, and W is the expected cycle time. Extensions and generalizations to this equation have been studied by Maxwell (1970) and Brumelle (1972). Maxwell's work first suggests that the average inventory level term, i.e., $L(t)$, can be generalized to any integrable functions, e.g., holding cost $L_g(t)$ associated with WIP, which then results in a functional dependent queueing equation, $L_g = \lambda W_g$. Brumelle's work gives a more complementary treatment of this relation, proposing a more general formula, $H = \lambda G$, where H and G contain moment information of the expected number of waiting customers and their waiting times respectively, denoted as L and W in the original formula. Later, solid proof for these types of relations was provided by Heyman and Stidham (1980) as well as Glynn and Whitt (1989).

Little's law (and its extensions) are very general and fit many queueing systems, but this also means it contains a certain level of generality and may not necessarily provide precise knowledge that is required to govern an optimization task, especially for the case of RLM systems. Because of the feedback loop and competitive nature, reducing the inventory level in one particular buffer in a RLM system, say L_i, will usually block the service in another and thus lead to an increase in buffer level, say L_j, if L_i and L_j compete for the same resources; this further affects the inventory level and work flow in between. We will discuss two basic cost realizations through departing from the Little's law, as depicted in Figure 1, tailored to fit the structure of RLM systems.

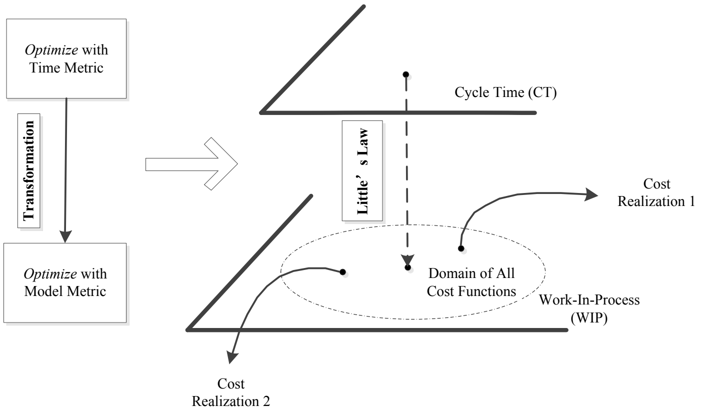

Figure 1: Cost Function Realization.

First, we denote each waiting line in an RLM system by a unique label $i \in \{1,2,...,l\}$, and let L_i be the current queue length in queue i, regardless of its network position and class type. Then, departing from Little's law, namely, $L = \lambda W$, we examine the following two cost realizations:

1. For a stable RLM network, minimize $\sum_{i=1}^{l} L_i$.
2. For a stable RLM network, minimize $\sum_{i=1}^{l} L_i^2$.

The first realization can be regarded as a direct generalization from the original Little's law to a multi-class queueing network (Wein 1988). The second one can be thought of as a quadratic functional of this kind, in the sense that it considers some second moment information of the buffer level. Under ideal conditions of an exponential queueing system, minimizing the first one above would lead to optimizing the cycle time. On the other hand, one would also expect that the second would lead to some reduction or improvement of the cycle time, though perhaps not optimal.

Below, we discuss some extensions of the conventional Markov decision process in order to accommodate multiple cost realizations.

2.2 Markov Decision Process and Some Extensions

Departing from Little's law motivates the use of multiple one-stage cost functions for the optimization process. In order to accommodate this and provide a flexible platform for comparing the induced control policy, we discuss extensions to the conventional definition of MDP.

Consider the conventional definition of an MDP with tuples $\{S,A,P,c\}$, where $s \in S$ represents the states, $a(s) \in A$ represents the action mapping from state to control, $p_{s,s'}(s,a) \in P$ are the transition probabilities and $c(\cdot)$ is the one-stage cost function independent of the next state. For a classic problem, e.g., a finite and recurrent chain, existing solution methods based on gradient and linear function approximation, e.g., TD(λ), are sufficient to solve this complex optimization problem. Usually, this results in a value function $J(\cdot)$, which is further used to differentiate between policies $\pi \in \Pi_{ad}$, where Π_{ad} is the admissible policy set for the underlying MDP problem. By convention, policies are compared through the value function

$J(\cdot)$ defined with the one-stage cost function c. In the case of the average cost (AC) criterion, this can be shown as: $J_c^{\pi}(s) = \lim_{N\to\infty} \frac{1}{N} E^{\pi}\{\sum_{k=0}^{N-1} c(s_k)|s_0 = s\}$. A preference over the policy, say, $\pi^a \succeq \pi^b$ is then defined with the partial order over the value function, e.g., $J_c^{\pi^a}(\cdot) \leq J_c^{\pi^b}(\cdot)$, and optimal policy π_c^* is one that minimizes the value function. One limitation for the conventional formulation is that it only allows the comparison of policies selected with respect to the same cost (function) realization or under some strict constraints. And when this one cost function is biased due to insufficient knowledge of the underlying system, the induced policy can be also biased and thus becomes overall non-optimal. Thus, in order to avoid the use of a biased cost function and accommodate comparison among multiple cost functions and their induced policies, we need the following extensions.

First, we denote $Q^{\pi}(\cdot)$ as a new metric corresponding to the system performance under a particular control policy π. For the control problem of RLM systems, the performance of $Q^{\pi}(\cdot)$ is different from the value function $J_c^{\pi}(\cdot)$, in the sense that the former corresponds to the time metric, e.g., cycle time, and the latter corresponds to the modeled metric, e.g., functionals of WIP. The preference over policy, say, $\pi^a \succeq \pi^b$ is then re-defined over the partial order of the new metric, e.g., $Q^{\pi^a}(\cdot) \leq Q^{\pi^b}(\cdot)$. And we have the new optimality over whole admissible policy set as: $\pi^* = \arg\min_{\pi\in\Pi_{ad}} Q^{\pi}(\cdot)$, which we denote as the exact (optimization) problem. It's known that this corresponds to the direct optimization of the RLM system in terms of the time metric, and we also refer to this as optimization in the policy space.

Due to the dimensionality and tractability issues, there are no efficient algorithms that can be used to track and converge along the unknown surface of the performance $Q^{\pi}(\cdot)$ for the exact problem. Alternatively, we say that this can be partially solved by exploiting the so-called un-dominated policies with simulation-based approximate dynamic programming methods. We denote an un-dominated policy as $\pi_{c_i}^*$, which is defined as $\pi_{c_i}^* \succeq \pi$ for any $\pi \in \Pi_{ad}$ with the conventional policy preference notation; in other words, it's an optimal policy for one particular cost function realization c_i. Then essentially, we try to convert the original exact problem, e.g., $\pi^* = \arg\min_{\pi\in\Pi_{ad}} Q^{\pi}(\cdot)$, to a more tractable searching problem among a set of induced un-dominated policies $\{\pi_{c_1}^*, \pi_{c_2}^*, ..., \pi_{c_i}^*, ...\}$, where each single policy in this set can be efficiently solved with an existing tractable algorithm, e.g., TD(λ), and we denote the procedures for finding each one of the un-dominated policies as optimization in the value space. One known practical issue with this framework is its potential for a large or infinite search space over the un-dominated policy set. This, however, needs to be studied independently, and is beyond the scope of this paper. In the rest of this study, we restrict the set of un-dominated policies to be a small finite set corresponding to our heuristically selected cost functions to serve the purpose of this study.

In Section 3, we'll talk about the detailed model of Intel Mini-Fab that is used in this study.

3 SIMULATION MODEL

In this section, we will discuss the simulation model and its key features. The model used throughout this paper is the so-called Intel Mini-Fab, which is a joint research effort between research labs from Arizona State University (ASU) and Intel Corporation. The model is a benchmark system for all kinds of semiconductor research, and also is a popular one used for evaluating different sequencing rules.

The Intel Mini-Fab system has three stations, diffusion, ion implantation and lithography, denoted as Station 1, 2, and 3 respectively. Both the diffusion and ion implantation stations have two machines (A, B) and (C, D) that can do parallel jobs, while the lithography station has only one machine (E). A diagram of the general structure of the Mini-Fab is given in Figure 2. The manufacturing process follows the sequence of the so-called six steps marked with arrows in Figure 2 with one re-entry to each of its stations. An additional 9 buffers for batching processes before buffers 1 and 5 are used; details of the batching rule and buffer settings can be found in (Kempf 1994).

The model also includes features like multiple products, a diffusion-like batching process, machine failure, preventive maintenance work, as well as other important factors like modeling of operators, setup times, machine loading/unloading, and transportation. The Mini-Fab operates 24 hours, 7 days a week, with each 12 hours as a shift. In each week, the nominal arrival rate is 84 lots per week, with 60.7%

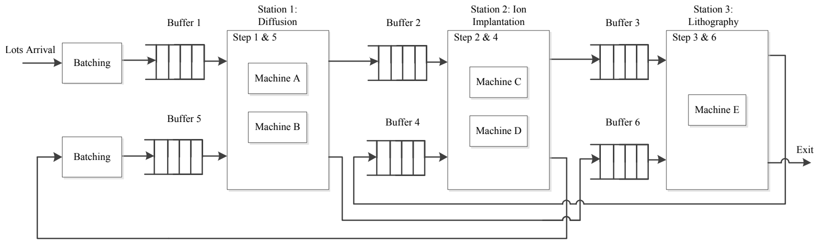

Figure 2: Intel Mini-Fab.

of the arrivals being product A, 35.7% are product B, and the rest are test wafer. Stochastic arrival and exponential processing times are used in our study, where the arrival is modeled as a Poisson process. Arrivals then go into the first batching process, see Figure 2, and are released into the system only if there is room in buffer 1. The detailed processing time as well as loading/unloading time, denoted as L/U below, are given in Table 1 for each step in the manufacturing process.

Table 1: Time Specification for Intel Mini-Fab Model.

Machines	Processing Steps	Processing Time (min)	L/U (min)
A & B	1 & 5	Step 1: 225, Step 5: 255	20/40
C & D	2 & 4	Step 2: 30, Step 4: 50	15/15
E	3 & 6	Step 3: 55, Step 6: 10	10/10

In the Mini-Fab model, machines C and D are considered to have random failures, which occur with a uniform distribution between 24 to 76 hours. After a machine fails, emergency repair work will be provided, which also takes a uniform distribution between 360 to 480 minutes for a technician to fix. In addition to the unexpected machine failure, the Mini-Fab model considers scheduled down time to provide preventive maintenance to all stations and machines. These schedules are deterministic and repeated every day or shift, where the detailed schedule can be found in Table 2.

Table 2: PM Schedule.

Machine	PM Schedule (Starting Time)	PM Duration (min)
A & B	12 a.m. & 12 p.m.	75
C & D	12 a.m/p.m. & 2 a.m/p.m.	120
E	3 a.m/p.m.	30

4 SIMULATION-BASED APPROXIMATE DYNAMIC PROGRAMMING APPROACH

In this section we discuss the simulation-based ADP approach utilized for finding un-dominated polices for sequencing control of the Mini-Fab system. We first discuss the Markov model of this RLM system and its optimization criteria, and we then focus on details and the integration of different cost realizations within our newly proposed extended actor-critic architecture.

4.1 Underlying Markov Model

We take a general form of a Markov model with tuples $\{T,S,A,P,c\}$ to represent the system, where T represents the time epochs, S represents the state space, A is the control space, P is the transition matrix induced by the controlled Markov process with each of its elements to have the form $p(\cdot|s,a)$, and $c(\cdot)$ is the cost function we'll discuss later. For practical purposes, the state and transition probability of the Mini-Fab are approximated with an ideal exponential queue of the same network structure, and all other non-exponential and random events are not explicitly considered, e.g., deterministic maintenance, operators' availability and machine failure. Hence the state is defined as follows:

$$s(t) := (L(t), V(t), B(t)), \forall t \in T, \tag{1}$$

where $L(t)$ contains an inventory level of six main buffers at time t with $L(t) := (l_1(t), l_2(t), \dots, l_6(t))^T$, and $V(t)$ contains the six pre-batching buffer levels at time t with $V(t) := (v_1(t), v_2(t), \dots, v_6(t))^T$, and $B(t)$ is the inventory level at the three batching processes at time t with $B(t) := (b_1(t), b_2(t), b_3(t))^T$.

In the Intel Mini-Fab model, the control $a \in A$ only considers the sequencing decision for the six main buffers as follows:

$$a(s) := (a_1(s), a_2(s), a_3(s), a_4(s), a_5(s), a_6(s)), \tag{2}$$

where each $a_i(s)$ is a binary decision variable, with value one representing selection of the corresponding buffer to be served next, and value zero representing not selected. To reduce the complexity and stabilize the controlled system further, we assume that all controls $a(s)$ belong to a certain constraint subset $\mathscr{A} \subset A$ for the RLM system. One of the constraints between the control of competing buffers is called the non-idling policy. Before we specify the details of the constraint set, we first partition the machines and buffers into different sets according to their stations. Then, we have three machine sets $\mathbf{M}_1, \mathbf{M}_2, \text{and}\, \mathbf{M}_3$, corresponding to station 1, 2, and 3. And we have $\{M_a, M_b\} \subset \mathbf{M}_1, \{M_c, M_d\} \subset \mathbf{M}_2, \text{and}\, \{M_e\} \subset \mathbf{M}_3$. Furthermore, we also partition buffers and control action by following the same rule. Then we have $\{a_1, a_5\} \subset \mathbf{A}_{\mathbf{M}_1}, \{a_2, a_4\} \subset \mathbf{A}_{\mathbf{M}_2}, \text{and}\, \{a_3, a_6\} \subset \mathbf{A}_{\mathbf{M}_3}$. The non-idling policy can be visualized as if there is at least one machine available in the downstream station, then one of the holding buffers will release the lots as: $\sum_{a_i \in \mathbf{A}_{\mathbf{M}_k}} a_i = \mathbf{1}_{\{\mathbf{M}_k = \text{Idle}\}}, \forall i \in 1,2,...,6,$ and $k \in 1,2,3$. In addition, the so-called anti-block mechanism is needed in order to stabilize the controlled reentrant line. The anti-block mechanism only allows the buffer to release the lot to the available machine when there is additional space at the buffer in the next processing step, which can be seen as: $a_i \leq \mathbf{1}_{\{L_{next\ step} < L_{capacity}\}}, \forall i \in 1,2,...,5.$

4.2 One Stage Cost Functions and Average Cost Criteria

Following our discussion of the cost realizations in Section 2.1, we first introduce the following two basic one-stage cost functions:

1. $c_1(s) := \mathbf{L}^T \cdot \mathbf{c}_l + \mathbf{V}^T \cdot \mathbf{c}_v + \mathbf{B}^T \cdot \mathbf{c}_b$
2. $c_2(s) := \mathbf{L}^T \mathbf{C}_l \mathbf{L} + \mathbf{V}^T \mathbf{C}_v \mathbf{V} + \mathbf{B}^T \mathbf{C}_b \mathbf{B}$

where $\mathbf{L}, \mathbf{V}$, and $\mathbf{B}$ are state vectors. Cost coefficients $\mathbf{c}_l$, $\mathbf{c}_v$, $\mathbf{c}_b$, and $\mathbf{C}_l$, $\mathbf{C}_v$, $\mathbf{C}_b$ are in vector and matrix form that correspond to each of the buffers and their inventory levels. The first cost function is an L_1 norm-like function, which only considers the total buffer level at each time epoch. Coefficients are set to be one, except for those holding batched products, e.g., buffers 1 and 5. This cost function is a direct relaxation of Little's law to multi-class queueing network; it tries to minimize the total inventory regardless of the network structure. The second one considers the quadratic functional of each buffer level, and as a second realization, this one will also lead to the reduction in cycle time. Besides, the moment information it contains may also lead to a lower variance compared to the first.

With the underlying Markov model and cost functions, we now discuss the optimization criteria for the process. We assume appropriate *uniformization* of the original continuous model, and only discuss its discrete counterpart in the rest of this paper. Focusing on long run steady-state performance, we then formulate this optimization process with the average cost criterion:

$$J^{\pi}(s) = \lim_{N \to \infty} \frac{1}{N} E^{\pi} \{ \sum_{k=0}^{N-1} c(s_k) | s_0 = s \}, \tag{3}$$

where $J^{\pi}(s)$ is the average cost under the policy π, and for simplicity we omit the subscript c in value function for the rest of the paper. The optimal $J^{\pi^*}(s)$ admits the minimal value of the average cost over all policies with respect to the given cost realization c over the infinite time horizon, and we denote this by $\mu^*(s) = J^{\pi^*}(s)$. Furthermore, it's assumed that the *weak accessibility* (Bertsekas 2007) condition holds, and we have $\mu^*(s) = \mu^*$ for all initial states s. Then, we consider the following average cost optimality equation (ACOE): $\mu^* + h^*(s) = \min_{a(s)\in\mathscr{A}(s)}\{c(s) + \sum_{s'} p(s'|s,u)h^*(s')\}$, where μ^* is the optimal average cost and $h^*(s)$ is the differential cost between the current cost $c(s)$ and the optimal average cost μ^*. The optimization process will then be running iteratively via the so-called extended actor-critic architecture discussed next.

4.3 Extended Actor-Critic Architecture and TD(λ)

Different from the conventional actor-critic, the new architecture integrates our newly defined performance measure and cost realizations into the existing framework. The new architecture involves two coupled layers, policy space and value space, also denoted as outer layer and inner layer respectively; see Figure 3. We assume from this point that the cost functions are parameterized by α, i.e., c_α. Recalling our discussion in Section 2.2, we consider the outer layer as a generalized problem that corresponds to optimization in the whole admissible policy space, e.g., $\pi^* = \arg\min_{\pi\in\Pi_{ad}} Q^\pi(\cdot)$. Then, by some mechanisms not specifically detailed at this point, one could have updates of c_α based on the performance $Q(\alpha)$ so as to minimize overall performance. The inner layer corresponds to the learning and optimization process with the conventional MDP formulation. We denote this new architecture as the *extended actor-critic architecture*. Different from the conventional one-shot optimization of the inner layer with only one choice of the cost function, the new outer layer now offers a flexible platform for the performance comparison, and thus provides more insight to help select the cost function. As discussed earlier, a full feedback loop for the outer layer may include extensive search in large or infinite space, so in this paper, we restrict this feedback to provide only simple (heuristic) options (see the discussion in Section 5), that leads to the reduction in cycle time.

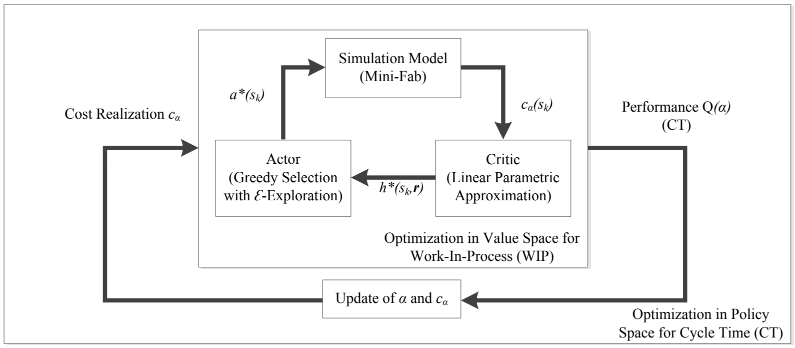

Figure 3: Extended Actor-Critic Architecture.

To solve the MDP problem efficiently and find all those un-dominated policies, the inner loop plays an important role, and we'll now focus on this part. In particular, we introduce the convergent approximate method, i.e., the TD(λ) algorithm as our inner layer. This algorithm is studied extensively by Sutton (1988), and when coupled with the gradient method and linear function approximation, solid convergence analysis has been reported by Tsitsiklis and Van Roy (1999). First, we consider the following approximated version of ACOE:

$$\tilde{\mu}^* + \tilde{h}^*(s,\mathbf{r}) = \min_{a(s)\in\mathscr{A}(s)}\{c(s) + \sum_{s'} p(s'|s,a)\tilde{h}^*(s',\mathbf{r})\}, \tag{4}$$

where $\tilde{h}^*(s,\mathbf{r})$ corresponds to an approximation to the true differential cost $h^*(s)$. We restrict this approximation to be linear, e.g., $\tilde{h}^*(s,\mathbf{r}) = \phi^T(s)\cdot\mathbf{r}$, with a set of basis function: $\phi(s) = (\phi_1(s)\,\phi_2(s)\cdots\phi_n(s))^T$. The basis function here could be selected as simple functions of each buffer level, which also depends on the type of the cost function used. Then, $\tilde{\mu}^*$ and $\tilde{h}^*(s,\mathbf{r})$ can be found with the inner layer, which contains three parts: a simulation model for generating sample trajectories, a critic part to approximate the differential function, and an actor part for making correct control actions.

For the critic part, the approximation to the differential cost $\tilde{h}$ is approximated in the subspace spanned by the feature vector, and a linear approximation structure is then used to ensure convergence, while the coefficients $\mathbf{r}$ are updated via the gradient-like method:

$$\mathbf{r}_{k+1} = \mathbf{r}_k + \gamma_k \cdot d_k \cdot \mathbf{z}_k, \tag{5}$$

where γ_k is the step size following standard convergence assumptions: $\sum_{k=0}^{\infty}\gamma_k = \infty$ and $\sum_{k=0}^{\infty}\gamma_k{}^2 < \infty$. $d_k := c(s_k) - \tilde{\mu}_k + \tilde{h}(s_{k+1},\mathbf{r}_k) - \tilde{h}(s_k,\mathbf{r}_k)$ is the temporal difference (TD) error defined as the difference between two consecutive approximations, and $\mathbf{z}_k := \sum_{t=0}^{k}\lambda^{k-t}\phi(s_t)$ is the eligibility trace. $\tilde{\mu}_k$ is the current approximation to the true average cost, which is updated with: $\tilde{\mu}_{k+1} := (1-\sigma_k)\cdot\tilde{\mu}_k + \sigma_k \cdot c(s_k)$.

For the actor part, the control policy is updated with an ε-greedy rule with respect to the current estimate:

$$a^*(s) = \arg\min_{a\in\mathscr{A}(s)}\{c(s,a) + \sum_{s'=1}^{n} p(s'|s,a)\tilde{h}(s',\mathbf{r}_k)\}. \tag{6}$$

This updating can be further simplified with the *structure property* of the RLM system reported by Ramirez-Hernandez and Fernandez (2007) as $a^*(s) = \arg\min_{a(s)\in\mathscr{A}(s)}\{a_1\cdot\tilde{\Delta}_1(s,\mathbf{r}_k) + \cdots + a_6\cdot\tilde{\Delta}_6(s,\mathbf{r}_k)\}$. Where $\tilde{\Delta}_i(s,r_k)$ is an approximation to the expected savings in optimal differential costs defined as $\tilde{\Delta}_i(s,\mathbf{r}_k) := \theta_i\cdot(\tilde{h}^*(B_i s,\mathbf{r}_k) - (\tilde{h}^*(s,\mathbf{r}_k))$; θ_i here represents the expected time rate required for the state transition from state s to $B_n s$ as a result of the control action $a_i = 1$, and $B_i s : \mathbb{R}^n \mapsto \mathbb{R}^n$ is defined as a controlled mapping between the current state and the next state.

We next present our empirical results with detailed performance comparisons, first for the two basic cost realizations, then for more general cases.

5 SIMULATION STUDY

In this section, we present our empirical results with respect to various cost functions. We first introduce some experimental settings, then we focus on the detailed performance obtained with both basic cost realizations as well as more general forms. The simulation model here and its experiments are built and conducted with the Arena simulation software (Kelton et al. 2010); in particular, the simulation-based ADP methods are coded using Arena's built-in VBA routine. During the course of the simulation, each cost function is tested for 200 replications of 9600 hours length each, with a total of 960000 hours learning and another 960000 hours of evaluation period. This number and length of replications provided uniformly good statistical precision across the outputs (95% confidence interval half widths within 3% of the respective sample means), and changing the cost functions significantly changes the performance. During the learning period, an ε-greedy updating method is used to provide additional trade-offs between exploiting and exploration over the state and control space. Some important parameters and settings are summarized below:

1. As the original RLM network with 84 lots per week is highly congested, we used a reduced arrival rate with 87% of the nominal arrival per week to present our numerical results better.
2. All scheduled or unscheduled repair and maintenance work will be finished regardless of the availability of the technician.
3. Machine setup times are considered to be negligible and not contribute to the cycle time.
4. λ for temporal different algorithm is chosen at 0.95.
5. σ_k and γ_k have the form of $\frac{1}{K_1+epoch}$ and $\sigma_k = K_2\gamma_k$, where K_1, K_2 are some real constants.
6. ε for exploration is updated at each replication with $\frac{5}{replication}$%.

In Table 3, we first list the detailed cycle time with a 0.95 confidence interval (CI) and standard deviation (SD) for each product type and with respect to two basic cost realizations. The cost realization c_2 achieves lower cycle times in all three products types, also with a slight advantage in standard deviation, as expected, since it contains some second moment information. In contrast, we list in Table 4 the aggregated measure of cycle time for all type of products combined, which is denoted as Q, and the corresponding optimal average cost, which is denoted as J^*.

Table 3: Detailed Performance of Two Cost Realizations under Reduced Arrival Rate.

	Cost Realization c_1		Cost Realization c_2	
	Cycle Time (Hour)	SD	Cycle Time (Hour)	SD
Product A	[238.57 250.13]	47.22	[237.01 247.57]	44.89
Product B	[219.19 226.19]	32.72	[218.98 224.72]	31.21
Test Wafer	[218.38 225.50]	32.40	[217.33 222.85]	31.71

Table 4: Measurement Comparison under Reduced Arrival Rate.

	Optimal J^*	Q
Cost Realization c_1	91.44	235.45
Cost Realization c_2	2971.34	233.86

Data in Table 4 give a direct comparison between the optimal value achieved with the two cost realizations considered. A key difference here as compared to the conventional MDP formulation is that the performance measure of Q in Table 4 provides a feasible platform to compare different cost functions directly and their induced control policies, which are not comparable under the conventional settings, e.g., comparison of J^*. Next, the detailed steady-state buffer levels at the six main buffers are summarized in Figure 4. Buffers 1, 2, and 3 are heavily loaded compared to their capacity levels, and buffers 4, 5, and 6 achieve low average buffer levels during the simulation.

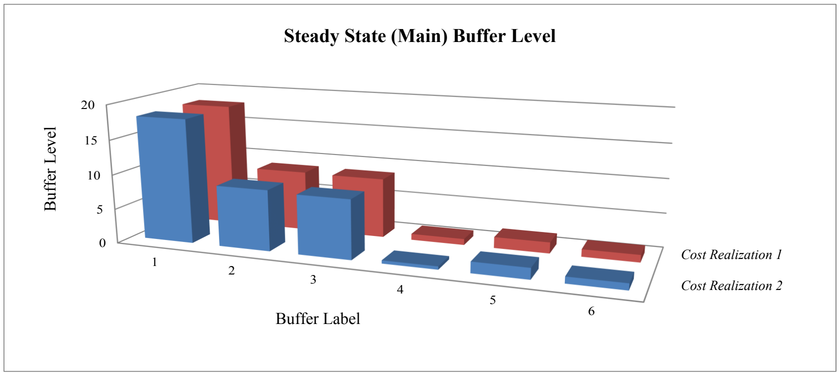

Figure 4: Steady State Buffer Levels for Two Basic Cost Realizations.

As we now have compared the two basic cost realizations and their induced control policy to the Mini-Fab system, a natural question that may arise is whether there is a better cost realization for this problem. We show below that these two simple cost realizations can be easily extended to general forms, in the sense that these two do not include any in-depth domain knowledge. A reasonable generalization is to adjust the cost function to have a weights-adjusted form, e.g., $\sum_{i=1}^{n} \theta_i f(L_i)$, where θ_i is some weighting function with or without certain constraints, and $f(L_i)$ are some functionals that depend on the buffer level L_i. In the following simulation studies, we use the weighted forms of both cost realizations to address this issue. In particular, we heuristically tested several different sets of weights that correspond to different user preferences over the state space with our new architecture. For simplicity, we only consider preference over two buffer classes. As previous results showed in Figure 4, the buffers can be categorized into two classes: heavily loaded, e.g., buffers 1, 2, and 3, and lightly loaded, e.g., buffers 4, 5, and 6. We then let α be the ratio between the the holding costs for these two classes, e.g., $\alpha = \frac{\theta_1}{\theta_5} = \frac{\theta_2}{\theta_4} = \frac{\theta_3}{\theta_6}$, while detailed holding costs within each class are kept the same following the discussion in Section 4.2. We consider several different choices of αs, with detailed results summarized in Table 5 and 6.

Table 5: Detailed Performance with Different Preference Ratio α.

	Weighted Cost Realizations c_1				
	Product A	Product B	Product TW	Q	$\frac{Q-Q_o}{Q_o}$[1]
$\alpha = 16$	[361.73 370.23]	[347.19 352.55]	[345.72 350.26]	359.32	-52.60%
$\alpha = 9$	[243.58 251.48]	[223.25 227.69]	[222.01 226.89]	238.46	-1.27%
$\alpha = 4$	[245.84 252.72]	[224.35 228.23]	[222.92 226.90]	239.78	-1.83%
$\alpha = 1$	[238.57 250.13]	[219.19 226.19]	[218.38 225.50]	235.45	0 %
$\alpha = 9/11$	[240.67 247.11]	[220.55 225.09]	[220.26 224.48]	235.25	+0.08%
$\alpha = 7/13$	[211.51 221.99]	[195.48 206.48]	[194.48 202.70]	209.91	+10.84%
$\alpha = 1/4$	[206.96 219.22]	[193.73 202.33]	[192.95 202.13]	206.89	+12.13%

Table 6: Detailed Performance with Different Preference Ratio α.

	Weighted Cost Realizations c_2				
	Product A	Product B	Product TW	Q	$\frac{Q-Q_o}{Q_o}$[1]
$\alpha = 16$	[364.62 372.06]	[348.71 353.05]	[346.47 351.57]	361.10	-54.40%
$\alpha = 9$	[361.97 369.43]	[346.46 352.16]	[344.02 350.14]	358.90	-53.46%
$\alpha = 2.25$	[365.30 372.38]	[348.33 353.83]	[346.99 352.73]	361.50	-54.57%
$\alpha \approx 1.493$	[238.91 252.03]	[230.29 236.19]	[219.42 225.14]	236.29	-1.03%
$\alpha = 1$	[237.01 247.57]	[218.98 224.72]	[217.33 222.85]	233.86	0%
$\alpha = 4/9$	[216.77 225.93]	[202.50 208.74]	[201.51 207.53]	214.95	+8.08%
$\alpha = 1/16$	[215.94 224.28]	[200.63 208.33]	[199.88 207.40]	213.75	+8.59%

[1] Q_o corresponds to a Q value with $\alpha = 1$.

The heuristics behind the choice of α can also be regarded as the preference for balancing different buffer levels between two classes of buffers. In particular, $\alpha = 1$ indicates that there is no preference for holding one inventory at either heavily loaded or lightly loaded buffers, while $\alpha > 1$ or $\alpha < 1$ indicate the preference for lowering the first or second class of buffers respectively. Essentially, lowering the second class can be thought to be similar to the priority sequencing policy, e.g., last-buffer-first-serve (LBFS), and lowering the first class to be another, named first-buffer-first-serve (FBFS). Different from those two heuristic policies, which gives highest priority based purely on network position, the use of α allows more flexible and delicate tuning of the buffer preference of the system. The data here show interesting results. First, direct application of the ADP method to sequencing control of the Mini-Fab with no preference, e.g., $\alpha = 1$, does not yield the best CT from either of the two basic cost realizations. Second, there is a range of α such that the performance is neither improved nor degraded, for example, $\alpha \in [\frac{9}{11}\ 9]$ for the first realization and $\alpha \in [1\ 1.493]$ for the second. There exists some certain threshold for which α goes beyond or below, where the performance then will substantially change. Also, although the variance data are not shown here, the weighted cost realization c_1 can achieve the lowest cycle time while the weighted realization c_2 always offers a lower variance as a trade-off.

The results thus confirm that the selection of the cost function for optimizing the RLM system can be sensitive and thus needs careful treatment and requires certain domain knowledge. Besides, the use of Little's law does provide a fundamental guideline for those problems, but it does not offer explicit and precise knowledge all the time. It also validates the use of the so-called extended actor-critic architecture as a complementary method in situations where only imprecise or incomplete knowledge is available.

6 CONCLUSION

We have presented our empirical study of implementing the MDP method with multiple one-stage cost functions for optimizing the sequencing control of a benchmark RLM system. In particular, we introduced multiple cost (function) realizations with the departing from the well-known queueing rule, Little's law, previously used to govern this kind of optimization process. We also provide certain modifications to the use of conventional MDP and a simulation-based ADP method with our proposed extended actor-critic architecture. The results show that the new architecture can generally offer more insight for the underlying process and provide comparable results. The study still leaves several interesting and open questions regarding the selection of one-stage cost functions in optimization and the use of simulation-based ADP methods. Further research could proceed in several directions, including the preference-assigning procedure to each buffer, adaptive cost realization, and selecting and comparing mechanisms for large or infinite realization sets.

REFERENCES

Bertsekas, D. P. 2007. *Dynamic programming and optimal control. Volume II*. 3rd ed. Bellmont, Massachusetts: Athena Scientific.

Brumelle, S. L. 1972. "A Generalization of $L = \lambda W$ to Moments of Queue Length and Waiting Times". *Operations Research* 20 (6): 1127–1136.

Choi, J.-Y., and S. Reveliotis. 2005. "Relative value function approximation for the capacitated re-entrant line scheduling problem". *IEEE Transactions on Automation Science and Engineering* 2 (3): 285–299.

Flores-Godoy, J.-J., W. Yan, D. W. Collins, F. Hoppensteadt, and K. Tsakalis. 1998. "A Mini-FAB simulation model comparing FIFO and MIVP(R) schedule policies (outer loop), and PID and H^{∞} machine controllers (inner loop) for semiconductor diffusion bay maintenance". In *Proceedings of the 24th Annual Conference of the IEEE on Industrial Electronics Society (IECON '98)*, 253–258.

Glynn, P. W., and W. Whitt. 1989. "Extensions of the Queueing Relations $L = \lambda W$ and $H = \lambda G$". *Operations Research* 37 (4): 634–644.

Heyman, D. P., and S. Stidham. 1980. "The Relation between Customer and Time Averages in Queues". *Operations Research* 28 (4): 983–994.

Kelton, W. D., R. P. Sadowski, and N. B. Swets. 2010. *Simulation with Arena*. 5th ed. New York, New York: McGraw-Hill.

Kempf, K. 1994. "Intel Five-Machine Six Step Mini-Fab Description". Accessed Feb. 1, 2011. http://www.fulton.asu.edu/aar/research/intel/papers/fabspec.html.

Kumar, P. R. 1993. "Re-entrant lines". *Queueing Systems: Theory and Applications* 13:97–110.

Kumar, P. R. 1994. "Scheduling semiconductor manufacturing plants". *IEEE Control Systems Magazine* 39 (11): 33–40.

Kumar, S., and P. R. Kumar. 2001. "Queueing network models in the design and analysis of semiconductor wafer fabs". *IEEE Transactions on Robotics and Automation* 17 (5): 548–561.

Little, J. 1961. "A Proof for the Queuing Formula: $L = \lambda W$". *Operations Research* 9 (3): 383–387.

Lu, S. H., and P. R. Kumar. 1991. "Distributed scheduling based on due dates and buffer priorities". *IEEE Transactions on Automatic Control* 36 (12): 1406–1416.

Maxwell, W. L. 1970. "On the Generality of the Equation $L = \lambda W$". *Operations Research* 18 (1): 172–174.

Powell, W. B. 2007. *Approximate dynamic programming: Solving the curses of dimensionality*. Hoboken, New Jersey: Wiley- Interscience.

Puterman, M. L. 1994. *Markov decision processes: Discrete stochastic dynamic programming*. New York, New York: John Wiley & Sons, Inc.

Ramirez-Hernandez, J. A., and E. Fernandez. 2007. “Control of a re-entrant line manufacturing model with a reinforcement learning approach”. In *Proceedings of the Sixth International Conference on Machine Learning and Applications (ICMLA’07)*, 330–335. Cincinnati, Ohio.

Ramirez-Hernandez, J. A., and E. Fernandez. 2009, December. “A simulation-based Approximate Dynamic Programming approach for the control of the Intel Mini-Fab benchmark model”. In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, 1634–1645. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Ramirez-Hernandez, J. A., and E. Fernandez. 2010. “Optimization of Preventive Maintenance Scheduling in Semiconductor Manufacturing Models Using a Simulation-Based Approximate Dynamic Programming Approach”. In *Proceedings of the 49th IEEE Conf. on Decision and Control*, 3944–3949. Atlanta, GA.

Sutton, R. S. 1988. “Learning to predict by the methods of temporal differences”. *Machine Learning* 3:9–44.

Tsakalis, K. S., J.-J. Flores-Godoy, and A. A. Rodriguez. 1997. “Hierarchical modeling and control for re-entrant semiconductor fabrication lines: a mini-fab benchmark”. In *Proceedings of the 6th international conference on emerging technologies and factory automation*, 508–513.

Tsitsiklis, J., and B. Van Roy. 1999. “Average cost temporal-difference learning”. *Automatica* 35:1799–1808.

Uzsoy, R., C. Lee, and L. A. Martin-Vega. 1994. “A review of production planning and scheduling models in the semiconductor industry part II: Shop-floor control”. *IIE Transactions* 26 (6): 44–55.

Vargas-Villamil, F. D., D. E. Rivera, and K. G. Kempf. 2003. “A hierarchical approach to production control of reentrant semiconductor manufacturing lines”. *IEEE Transactions on Control Systems Technology* 11 (4): 578–587.

Wein, L. M. 1988. “Scheduling semiconductor wafer fabrication”. *IEEE Transactions on Semiconductor Manufacturing* 1:115–130.

Whitt, W. 1984. “Open and closed models for networks of queues”. *AT & T Bell Laboratory Technical Journal* 63:1911–1978.

AUTHOR BIOGRAPHIES

XIAOTING CHEN is a Ph.D. candidate at the School of Electronic & Computing Systems at the University of Cincinnati. His current research area includes Markov decision processes, approximate dynamic programming, and simulation-based optimization methods. His e-mail address is chenxt@mail.uc.edu.

EMMANUEL FERNANDEZ is an Associate Professor in the School of Electronics & Computing Systems at the University of Cincinnati. His research areas of expertise are stochastic models, stochastic decision and control processes, and mathematical and computational operations research. His interests in applications are broad, spanning across the areas of manufacturing, operations and management, telecommunications, logistics, algorithms and software/Internet tools. He is a member of INFORMS, SIAM, and a Senior member of IEEE and IIE. His email address for these proceedings is emmanuel.fernandez@uc.edu.

W. DAVID KELTON is a Professor in the Department of Operations, Business Analytics and Information Systems at the University of Cincinnati, and Director of the MS - Business Analytics Program there. His research interests and publications are in the probabilistic and statistical aspects of simulation, applications of simulation, and stochastic models. He is co-author of *Simulation with Arena*, and was also coauthor of the first three editions of *Simulation Modeling and Analysis*, both for McGraw-Hill. He was Editor-in-Chief of the *INFORMS Journal on Computing* from 2000-2007. He served as WSC Program Chair in 1987, General Chair in 1991, was on the WSC Board of Directors from 1991-1999, and is a Founding Trustee of the WSC Foundation. He is a Fellow of both INFORMS and IIE. His email address is david.kelton@uc.edu.