Relative Value Function Approximation for the Capacitated Re-entrant Line Scheduling Problem

Jin Young Choi and Spyros Reveliotis
School of Industrial & Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332
{choijy68,spyros}@isye.gatech.edu

***Abstract*—The problem addressed in this work is that of determining how to allocate the workstation processing and buffering capacity in a *capacitated* re-entrant line to the job instances competing for it, in order to maximize its long-run / steady-state throughput, while maintaining the logical correctness of the underlying material flow, i.e., deadlock-free operations. An approximation scheme for the optimal policy that is based on *Neuro-Dynamic Programming* theory is proposed, and its performance is assessed through a numerical experiment. The derived results indicate that the proposed method holds considerable promise for providing a viable, computationally efficient approach to the problem, and highlight directions for further investigation.**

***Note to Practitioners*—Sequencing and scheduling problems arising in the context of contemporary manufacturing environments are known to be extremely hard. For this reason, in most practical situations, these problems have been resolved through the application of a number of *heuristics* – i.e., "rules of thumb" that are expected to provide reasonable performance. Things are complicated even further in the automated versions of these environments, since the applied sequencing and scheduling logic must guarantee, in addition to good performance, logically correct and smooth operation. Both, the logical and the performance-oriented control problem of flexibly automated production systems can be – and have been – addressed through formal systems theory. However, a challenging remaining problem is the approximation of the derived optimal policies in a way that will maintain near-optimality, and at the same time, it will be computationally tractable in the context of the "real-world" applications. Our past work has addressed this approximation problem primarily with respect to the optimal logical control policy. The work presented in this paper undertakes the complementary problem of approximating the optimal scheduling policy. To this end, we employ some recently emerged results from a field known as *Neuro-Dynamic Programming* (essentially, a systematic approximation framework for *Dynamic Programming*). Our results indicate that the proposed approximation framework holds considerable promise towards developing a systematic analytical methodology for deriving near-optimal and logically correct scheduling policies for flexibly automated production systems. More specifically, it is shown that, when applied to some prototypical problems concerning the scheduling of re-entrant lines with finite buffering capacity at their workstations, the proposed approximation framework (i) effectively integrates past results concerning the logical control of these environments, and (ii) the obtained performance is consistently superior to the performance provided by the typically used heuristics.**

***Index Terms*— Capacitated re-entrant line, Neuro-Dynamic Programming, relative value function approximation, scheduling.**

I. Introduction

In its basic definition [1], the *re-entrant line* consists of L workstations, W_1, W_2, $\cdots$, W_L, that support the production of a single item. Each workstation W_i, $i = 1, 2, \cdots, L$, possesses S_i identical servers, and the production of each unit occurs in M stages, J_1, J_2, $\cdots$,J_M; each stage J_j, $j = 1, 2, \cdots, M$, is supported by one of the system workstations, to be denoted by $W(J_j)$. Also, $M > L$, which characterizes the re-entrant nature of the line. The *capacitated* re-entrant line (CRL) [2], considered in this work, further assumes that each workstation has B_i buffer slots; each part visiting the workstation for the execution of some processing stage is allocated one unit of buffering capacity, which it holds exclusively during its entire sojourn in the station, while blocking other parts coming into the station. Once in the station, the part competes for one of the station servers for the execution of the requested stage. Moreover, the part maintains hold of its allocated buffer slot while being processed. After having finished the processing of its current stage at a certain station, the part waits in its allocated buffer for transfer to the next requested station. Due to the finite buffering capacity, this transfer should be authorized by a *structural control policy (SCP)* [3] ensuring that (i) the destination workstation has available buffering capacity, and (ii) the transfer is *safe*, i.e., it is still physically possible from the resulting state to process all running jobs to completion.

In the context of this operational framework, the problem considered in this work can be posed as determining how to allocate the workstation processing and buffering capacity to the competing parts, in order to maximize the long-run system throughput, while maintaining logical correctness of the material flow, i.e., deadlock-free operation. To facilitate the subsequent developments, it is further assumed that: (i) there exists an infinite amount of raw material waiting for processing at the line's Input/Output (I/O) station;[1] (ii) the processing time of stage J_j, $j = 1, 2, \cdots, M$, is exponentially distributed with finite non-zero rate μ_j;[2] (iii) the involved job transfer times are negligible when compared to the processing

[1]This assumption reflects the fact that we are interested in the optimization of the long-run system throughput.

[2]More general cases can be covered through approximation based on phase-type distributions.

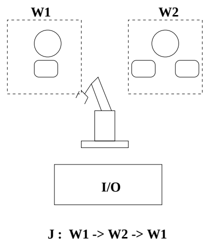

Fig. 1. Example: A capacitated re-entrant line

times.[3]

Example As a more concrete example consider the capacitated re-entrant line depicted in Figure 1. This line has two workstations, W_1, W_2, with S_1 = S_2 = 1 and B_1 = 1, B_2 = 2. The supported production sequence is $J = < J_1$, J_2, $J_3 >$, with $W(J_1)$ = $W(J_3)$ = W_1 and $W(J_2)$ = W_2. Stage processing times are exponentially distributed with rate μ_j, $j = 1, 2, 3$, and so are the involved transfer times, with a uniform rate $\alpha \rightarrow \infty$. For this small configuration, it can be easily seen that the system material flow will be deadlock-free as long as

$$|J_1| + |J_2| \leq B_1 + B_2 - 1 = 2, \tag{1}$$

where $|J_j|$, $j = 1, 2, 3$, denotes the number of job instances in $W(J_j)$ executing stage J_j. Under this condition, the CRL scheduling problem is to find an optimal control policy that maximizes the long-run system throughput by allocating the workstation processing and buffering capacity to the competing parts. $\triangle$

During the last fifteen years, there has been a significant number of works dealing with the scheduling problem in the original, uncapacitated re-entrant line; indicatively, we mention those published in [1], [4], [5], [6], [7], [8], [9]. However, the results derived in these past works cannot be immediately transferred to the capacitated re-entrant line model, due to the complications arising from the blocking effect taking place in this new environment. Characteristically, the work of [10] demonstrated through a simple example that these additional material flow dynamics negate in a strong qualitative sense prior analytical results obtained through the study of the basic re-entrant line model, and necessitate the re-examination of the problem in this new operational context. Motivated by these remarks, the work of [2] developed a formal approach for the analysis and control of capacitated re-entrant lines, based on the modelling framework of Generalized Stochastic Petri Nets (GSPN). This framework (i) allowed the seamless integration of logical/structural and timed-based aspects of the system behavior, (ii) provided an analytical formulation for the underlying scheduling problem, and (iii) led to some interesting qualitative insights regarding the structure of the optimal scheduling policy. However, the practical applicability of the framework of [2] to "real-world" applications is restricted by the fact that it requires the complete enumeration of the underlying state space, the size of which is a super-polynomial function of the elements defining the considered CRL scheduling problem.

[3]This assumption is quite representative of the operations taking place in highly integrated manufacturing systems, where a material handling device like a central robotic manipulator rapidly transfers jobs among a set of workstations located around it. On the other hand, more general cases can still be addressed through the presented approach, by defining virtual workstations performing job transfer operations.

Currently we lack a formal characterization of the computational complexity of the optimal policy for the considered CRL scheduling problem. Yet, existing results on the complexity of the optimal deadlock avoidance for sequential resource allocation systems [11] and the optimal control of queueing networks [12] seem to suggest that this policy will be computationally intractable. Hence, there is an apparent need for scalable and efficient approximations to it. A computational framework that holds considerable promise for providing such scalable and efficient approximations to the optimal solution of the considered CRL problem is that of *Neuro-Dynamic Programming* (NDP) [13].

Of particular interest to this work are the so called *parametric representation* NDP methods [13], [14], [15], [16]; these methods recast the considered scheduling problem to the problem of selecting an appropriate set of values for a parametric architecture that will eventually define the adopted scheduling policy. Conceptually, the deployment of such an approach consists of two major steps: (i) the specification of the approximation architecture and its aforementioned parametrization, and (ii) the design of effective algorithms for tuning the parameters of the approximation architecture, when applied on any given CRL configuration. The first of the above two steps – i.e., the specification of the approximation architecture – is typically driven by the following considerations: Since the approximating architecture will be involved in the real-time computation determining the applied scheduling policy, it must be of low, preferably polynomial, computational complexity. On the other hand, this constraint on the architectural complexity, and, in particular, on the "degrees-of-freedom" provided by the architecture parameterization, can have an adversarial impact on the representational capability of the architecture, and the eventual quality of the generated approximations. In an effort to effectively resolve this dilemma, the scientific community has currently confined itself primarily in the study of *linear* approximation architectures, i.e., architectures which are structured as a *weighted sum* of some preselected *"feature"* functions defined on the underlying state space. These feature functions seek to capture important aspects of the system state, and their selection is driven by practical experience, insight and/or any formal results available for the considered problem. The second major step underlying the development of a parametric representation NDP approach – i.e., the design of effective parameter-tuning algorithms for the adopted approximation architecture – is also driven by a combined concern for approximation accuracy and computational efficiency. Generally speaking, scalable versions of these algorithms seek to tune the architectural parameters by employing a host of approximate dynamic programming and estimation theory methodologies on a set of sample paths of the system behavior that are generated through simulation. However, the implementational details of these algorithms and their convergence properties strongly depend on the problem at hand and the structure of the adopted approximation architecture.

Motivated by the above remarks, this paper seeks to investigate the first of the aforestated issues, i.e., the efficacy of the linear approximating architectures for providing scalable and efficient solutions to the capacitated re-entrant line scheduling problem. More specifically, a *feature*-based compact representation is used in order to generate an effective approximation of the optimal control policy, and a particular set of feature functions is suggested and evaluated through a numerical experiment. A side-product of the presented work is the complete characterization of the considered scheduling problem as a *Continuous Time Markov Decision Process (CT-MDP)* [17]; this problem characterization is complementary to the original characterization provided in [2], and it is instrumental for (i) the justification of the proposed approximating architecture, and (ii) the assessment of the quality of the generated approximating policies, in the context of the presented numerical experiment. The obtained results indicate that the considered parametric representation NDP approach, in general, and the proposed linear architecture, in particular, hold considerable promise for providing effective and computationally efficient approximations to the optimal CRL scheduling policy that consistently outperform the typically employed heuristics.

The rest of the paper is organized as follows: Section II employs the CT-MDP framework towards the formulation and analysis of the considered scheduling problem. Section III provides a formal characterization of the proposed approximation framework and the associated feature selection problem.[4] Section IV proposes a particular set of features and assesses the representational capability of the resulting architecture through a numerical experiment. Finally, Section V concludes the paper and highlights directions for future work.

II. A Markov Decision Process Model for the Capacitated Re-entrant Line Scheduling Problem

This section formulates the CRL scheduling problem as a Markov Decision Process (MDP). The derived model provides a rigorous characterization of the CRL operation and the optimal scheduling policy. In this way, it offers the qualitative insights and a benchmarking baseline for the subsequent development of scalable approximating scheduling methods based on the emerging theory of *Neuro-Dynamic Programming* (NDP) [13].

The induced Continuous Time Markov Decision Process A formal characterization of the behavior generated by the considered CRL is facilitated by the following definition of its *state*.

Definition 1: Let n_{jw}, n_{jp}, and n_{jb} be respectively the number of jobs at stage J_j, $j = 1, 2, \ldots, M$, that are waiting for processing, being processed, and waiting for transfer to their next processing stage. Then, under the assumptions of exponential processing times and zero-transfer times, the CRL state is defined by the (3M-1) dimensional vector $< n_{1w}$, n_{1p}, n_{1b}, n_{2w}, n_{2p}, n_{2b}, $\ldots$, n_{Mw}, $n_{Mp} >$.[5]

The set of *events* that can change the system state comprises: (i) the event corresponding to the *loading* of a new job instance to the first required workstation, (ii) the events corresponding to the *advancement* of an unblocked job instance having completed the execution of its current stage, to its next stage, or out of the system, in case that the completed stage is the last one, (iii) the events corresponding to the *start* of the processing of a job instance waiting at a certain workstation, and (iv) the events corresponding to the *completion* of the running processing stage of a job instance. Furthermore, the CRL operation can be described by using this set of events as follows: Scheduling decisions are exerted by the system controller every time that an active process instance completes its running processing stage. Each scheduling option constitutes a sequence of job loading and / or advancing events that are physically *feasible*, but also *admissible* by the applied SCP. In general, the optimal CRL scheduling policy can involve *deliberate idleness* of some of the system resources. Therefore, a scheduling decision that blocks any job from loading or further advancement can be a viable – in fact, the optimal – option, in certain cases; we shall refer to such an option as a "*do nothing*" control. It must also be noticed, however, that any throughput-maximizing scheduling policy must be *globally non-idling*, i.e., there must always be at least one job instance under processing in the system.

Based on the above description, and under the assumption of zero transfer times, CRL states that result from the completion of the processing of an active job instance and correspond to decision epochs, present zero sojourn times, and therefore, they are characterized as *vanishing*. The remaining states contain at least one job instance that is in processing, and therefore, they have a finite sojourn time that is determined by the "*exponential race*" for completion of all the job instances that are in processing; these states will be characterized as *tangible*. Let $\mathcal{S}_{\mathcal{T}}$ and $\mathcal{S}_{\mathcal{V}}$ denote respectively the set of tangible and vanishing states. Then, the scheduling problem of maximizing the (steady-state) throughput of the CRL can be formulated as the sequential decision making problem of finding a policy that maximizes the (time-)average reward accumulated by the process defined by the CRL transitions through the vanishing states in $\mathcal{S}_{\mathcal{V}}$. A non-zero reward is obtained every time that the performed transition can result to the completion of an active process instance, while the time corresponding to such a transition is variable and it depends on the originating vanishing state and the selected control. We can formalize the above discussion as follows:

For each state $i \in \mathcal{S}_{\mathcal{V}}$, there exists a set of controls, $U(i)$, that is feasible at state i and finite: More specifically, this set of controls corresponds to all the process and SCP-enabled

[4]In fact, this section provides also a parameter-tuning methodology for the presented approximating architecture, that is derived directly from the formal characterizations of the considered CT-MDP problem and the proposed approximation framework. However, the practical applicability of this methodology is limited by the fact that it employs a complete enumeration of the underlying state space. Its role and value for the presented work lies in the (i) provision of a formal characterization for the parameter-tuning problem, and (ii) the facilitation of the computation and evaluation of the approximating policies developed in the context of the presented numerical experiment.

[5]The zero-transfer times assumption, when combined with the throughput maximization objective, implies that any job completed in the last stage can be unloaded immediately, and therefore, the component n_{Mb}, representing the number of jobs blocked in the last job stage, is redundant.

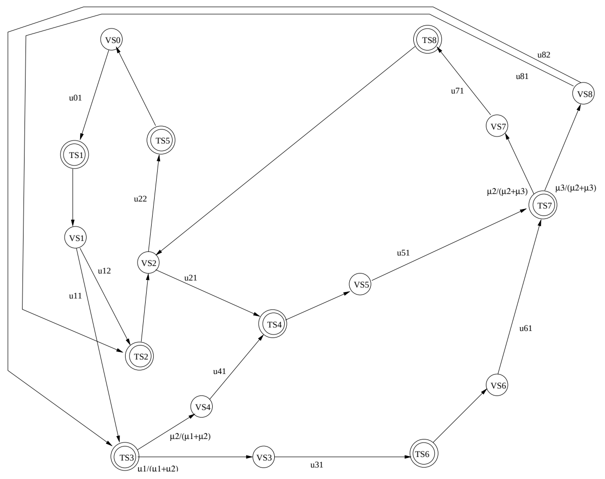

Fig. 2. Example: the induced CT-Markov process for the capacitated re-entrant line of Figure 1

TABLE I

EXAMPLE: VANISHING STATE INFORMATION FOR THE CT-MARKOV PROCESS INDUCED BY THE CAPACITATED RE-ENTRANT LINE OF FIGURE 1

VS_k	$n_{1w}n_{1p}n_{1b}$	$n_{2w}n_{2p}n_{2b}$	$n_{3w}n_{3p}$	resultant TS_k
0	0 0 0	0 0 0	0 0	(TS_1)
1	0 0 1	0 0 0	0 0	(TS_2,TS_3)
2	0 0 0	0 0 1	0 0	(TS_4,TS_5)
3	0 0 1	0 1 0	0 0	(TS_6)
4	0 1 0	0 0 1	0 0	(TS_4)
5	0 0 1	0 0 1	0 0	(TS_7)
6	0 0 0	1 0 1	0 0	(TS_7)
7	0 0 0	0 0 1	0 1	(TS_8)
8	0 0 0	0 1 0	0 0	(TS_2,TS_3)

event sequences that bring the system to a tangible state; we shall call such an event sequence a *clustered control*, and the resulting tangible state a *resultant tangible state*. Let $\Psi(i,u)$ be the *index set* of job instances being processed at the tangible state resulting from taking control $u \in U(i)$ at state i, and also, let $s(l) \in \mathcal{S}_\mathcal{V}$ denote the vanishing state resulting from the finishing of a job instance $l \in \Psi(i,u)$. Then, for $i,j \in \mathcal{S}_\mathcal{V}$, the *transition probability* $p_{ij}(u)$ is determined by

$$p_{ij}(u) = \frac{\sum_{l\in\Psi(i,u):s(l)=j}\mu_l}{\sum_{k\in\Psi(i,u)}\mu_k}. \tag{2}$$

The *sojourn time* associated with the transition resulting from the selection of control u at state i is exponentially distributed with *mean value*

$$\bar{\tau}_i(u) = \frac{1}{\sum_{k\in\Psi(i,u)}\mu_k}. \tag{3}$$

Let i_k be the system state at the k-th decision epoch t_k, and u_k the selected control at t_k. Then, $\{i_k, k = 0,1,\ldots\}$ is a *Continuous Time Markov Decision Process (CT-MDP)* with $\bar{\tau}_{i_k}(u_k) > 0$, where $\bar{\tau}_{i_k}(u_k)$ is the expected transition time resulting from applying control u_k at state i_k, $k = 0,1,2,\ldots$ Furthermore, the SCP logic applied during the system operation ensures that the system idle and empty state can be reached by any other process state, and therefore, the chain structure underlying this CT-MDP problem is strongly connected, or *communicating* in the relevant terminology [17].

TABLE II

EXAMPLE: RESULTANT TANGIBLE STATE INFORMATION FOR THE CT-MARKOV PROCESS INDUCED BY THE CAPACITATED RE-ENTRANT LINE OF FIGURE 1

TS_k	$n_{1w}n_{1p}n_{1b}$	$n_{2w}n_{2p}n_{2b}$	$n_{3w}n_{3p}$	next VS_k
1	0 1 0	0 0 0	0 0	(VS_1)
2	0 0 0	0 1 0	0 0	(VS_2)
3	0 1 0	0 1 0	0 0	(VS_3, VS_4)
4	0 1 0	0 0 1	0 0	(VS_5)
5	0 0 0	0 0 0	0 1	(VS_0)
6	0 0 0	1 1 0	0 0	(VS_6)
7	0 0 0	0 1 0	0 1	(VS_7,VS_8)
8	0 0 0	0 0 1	0 1	(VS_2)

Example Figure 2 presents the induced CT-Markov process for the CRL of Figure 1,[6] while the detailed characterization of the depicted states is provided in Tables I and II. Double-lined nodes in Figure 2 indicate the process *tangible* states, and the expressions on the edges emanating from them characterize the corresponding branching probabilities. Single-lined nodes are the *vanishing* states. $\triangle$

An algorithm for generating the state space of the considered CT-MDP Next we present an algorithm that generates the state space of the considered CT-MDP directly from the basic description of the system configuration. The proposed algorithm consists of two parts: (i) identifying the system *safe region*, i.e., this part of the state space from which it is physically possible to process all running jobs to completion without running into deadlock; (ii) generating the state space of the target CT-MDP, by starting from the null state and systematically exploring all possible clustered controls at every visited state, while using the information about the safe region for checking the structural admissibility of arising new states.

1. **Identifying the target safe region** The identification of the safe region for the buffer space allocation of the capacitated re-entrant line considered in this work is, in general, an NP-hard problem [11]. However, in [11], it is also shown that in many practical cases (e.g., when the capacity of a pertinently selected set of buffers is greater than 1), the problem can be resolved in polynomial time through one-step look-ahead deadlock detection. For the remaining cases, one can either (i) employ polynomial-complexity criteria / tests that will seek to identify a strongly connected *component* of the safe region that further contains the system empty state [3], or (ii) opt to ignore the complexity concern, and proceed to the identification of the entire safe region, by generating and trimming, with respect to the system initial empty state, the state transition diagram representing all the possible evolution of the buffer space allocation taking place in the underlying system; we refer the reader to [18], [3] for the algorithmic details.
2. **Generating the CT-MDP state space** After having obtained a characterization of the target safe region, the $\mathcal{S}_\mathcal{V}$ and $\mathcal{S}_\mathcal{T}$ state space are generated systematically as follows: For each vanishing state, all resultant tangible states are generated by enumerating all possible

[6]Notice that some control actions corresponding to clearly suboptimal decisions, resulting to unnecessary idling, were omitted during the development of the CT-MDP of Figure 2.

sets of clustered controls. These clustered controls are computed incrementally by augmenting generated subsequences of consecutive untimed controls until a resultant tangible state is reached. Then, for each resultant tangible state, the resulting vanishing states are generated, and this basic loop repeats itself. Details of the algorithm are as follows:

Algorithm to generate state space

(i) Let SR denote the set of states in the safe - more generally, admissible - region.

(ii) Initialize $\mathcal{S}_\mathcal{V}$ and $\mathcal{S}_\mathcal{T}$ by letting $\mathcal{S}_\mathcal{V} = \{s_0\}$ and $\mathcal{S}_\mathcal{T} = \emptyset$, where s_0 is the system empty and idle state.

(iii) If all states in $\mathcal{S}_\mathcal{V}$ are marked as "explored", then go to Step (vii). Otherwise, select one state from $\mathcal{S}_\mathcal{V}$, which is not explored, and mark it as "explored". Generate all resultant tangible states by (i) enumerating all possible sequences of untimed controls and (ii) checking if the resultant tangible states are in SR; if a resultant tangible state is not in SR, then remove it.

(iv) For each resultant tangible state, generate the vanishing states resulting from the completion of timed transitions, and put them into $\mathcal{S}_\mathcal{V}$, while avoiding duplication.

(v) Put the resultant tangible states generated in Step (iii) into $\mathcal{S}_\mathcal{T}$, while avoiding duplication.

(vi) Save the transitional information and go to Step (iii).

(vii) Done with $\mathcal{S}_\mathcal{V}$ and $\mathcal{S}_\mathcal{T}$ as the vanishing and tangible parts of the state space.

Systematic exploration of a vanishing state Step (iii) of the above algorithm can be readily supported through a Breadth-First-Search method. Furthermore, there are some pertinent observations that can lead to a more efficient enumeration of all clustered controls emanating from any given vanishing state. Most of these observations essentially constitute conditions under which a non-idling policy can be adopted without compromising optimality with respect to the considered performance objective of throughput maximization. Then, enforcing non-idleness reduces the number of viable controls at any vanishing state and leads to a smaller set of resultant tangible states.

Let us consider a clustered control $u_{s_v,k}$ at a vanishing state s_v, and let s_t be the resultant tangible state corresponding to $u_{s_v,k}$, where $s_t = < n_{1w}, n_{1p}, n_{1b}, n_{2w}, n_{2p}, n_{2b}, \ldots, n_{Mw}, n_{Mp} >$. We can characterize the "properness" of the control $u_{s_v,k}$ by investigating the "properness" of state s_t. For each workstation W_i, $i = 1, 2, \ldots, L$, let $\sigma(W_i)$ be the index set of job stages processed in workstation W_i, i.e., $W(J_j) = W_i$ for all $j \in \sigma(W_i)$. The following lemmas specify some conditions under which a non-idling policy is optimal.

Lemma 1: Under an optimal control policy, $1 \leq \sum_i \sum_{j\in\sigma(W_i)} n_{jp} \leq \sum_{i=1}^{L} S_i$.

Lemma 2: Under an optimal control policy, for each workstation W_i, $i = 1, 2, \ldots, L$, if $\sum_{j\in\sigma(W_i)}(n_{jw} + n_{jp}) = B_i$, then $\sum_{j\in\sigma(W_i)} n_{jp} = S_i$.

Lemma 3: Under an optimal control policy, for each workstation W_i, $i = 1, 2, \ldots, L$, if $n_{jw}+n_{jp} > 0$ for all $j \in \sigma(W_i)$, then $\sum_{j\in\sigma(W_i)} n_{jp} = \min\{S_i, \sum_{j\in\sigma(W_i)}(n_{jw} + n_{jp})\}$.

Lemma 4: Under an optimal control policy, for each workstation W_i, $i = 1, 2, \ldots, L$, if (i) $\sum_{j\in\sigma(W_i)} (n_{jw} + n_{jp} + n_{jb}) < B_i$, and (ii) $n_{j-1,b} + n_{jw} + n_{jp} > 0$ for all $j \in \sigma(W_i)$ (notice that $n_{0b} > 0$ always, by the assumption of an infinite WIP level waiting in front of the line), then $\sum_{j\in\sigma(W_i)} n_{jp} = \min\{S_i, \sum_{j\in\sigma(W_i)}(n_{j-1,b} + n_{jw} + n_{jp})\}$.

Lemma 5: Under an optimal control policy, for each workstation W_i, $i = 1, 2, \ldots, L$, if (i) $n_{j'-1,b} + n_{j',w} + n_{j',p} > 0$ for $j' = \arg\min_j\{j : j \in \sigma(W_i)\}$ and (ii) $n_{j',b} + \sum_l \sum_{\{k:k\in\sigma(W_l),k>j'\}} (n_{kw} + n_{kp} + n_{kb}) = 0$ (notice that $n_{Mb} = 0$ always, by the assumption of zero-transfer times), then $n_{j',p} = \min\{S_i, n_{j'-1,b} + n_{j',w} + n_{j',p}\}$.

Lemma 6: Under an optimal control policy, for each workstation W_i, $i = 1, 2, \ldots, L$, if (i) $|\sigma(W_i)| = 1$, (ii) $\sigma(W_i) = \{j \neq 1\}$ and (iii) $n_{jw} + n_{jp} + n_{jb} < B_i$, then $n_{j-1,b} = 0$.

Lemma 7: A control action that just loads a new job into $W(J_1)$ and does not start processing is redundant.

The formal proofs of Lemmas 1 – 7 can be based on the non-conflicting nature of the implied operations and can be established through techniques and arguments similar to those presented in [19](Chapter 3). Here we shall provide a more intuitive justification of their correctness. Hence, Lemma 1 addresses the non-optimality of a globally idling policy for the throughput maximization problem. Lemma 2 describes the optimality of a local non-idling policy for a workstation with its buffer full of non-completed job instances. Lemma 3 expresses the optimality of a local non-idling policy for a workstation which contains at least one non-completed job instance for each job stage supported by the workstation. Lemma 4 further generalizes Lemma 3 for the case that the workstation demonstrates free capacity; notice that in this case, the non-idleness enforcing condition counts also all the completed job instances waiting to enter the considered workstation. Lemma 5 states the optimality of a local non-idling policy for a workstation that contains at least one non-completed job instance to be processed at the earliest job stage in that workstation, while there are no job instances at all subsequent job stages to completion in the system. Lemma 6 applies to a workstation which processes only one job stage and indicates that any free buffering capacity on such a workstation should be immediately allocated to a requesting process. Finally, Lemma 7 results from the assumption of zero-transfer times, which allows a loading control to be performed only when the new loaded job is going to be started instantly after being loaded. The conditions expressed by Lemmas 1 – 7 must be integrated in the logic of any search procedure supporting the execution of Step (iii) of the state space-generating algorithm.

Characterizing the objective function of the CT-MDP In the CT-MDP framework, the long-run CRL throughput is modelled by the (time-)*average reward* to be accumulated by the considered process, formally defined by

$$\lim_{N\to\infty} \frac{1}{E\{t_N\}} E\{\int_0^{t_N} g(i(t), u(t))dt\}. \qquad (4)$$

In Equation 4, $g(i(t), u(t))$ is the *reward* per unit time obtained by taking control $u(t)$ at state $i(t)$ at time t; in particular, $i(t) = i_k$ and $u(t) = u_k$ for $t_k \leq t < t_{k+1}$. In the considered problem context, $g(i, u)$ is defined by:

$$g(i,u) = \begin{cases} \mu_M & \text{if the resultant tangible state has a job being processed in the last stage } J_M \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The *single-stage expected reward*, $G(i, u)$, corresponding to state i and control u, is given by

$$G(i,u) = g(i,u)\bar{\tau}_i(u) \tag{6}$$

From Equations 5, 6, we have

$$G(i,u) = \begin{cases} \mu_M \bar{\tau}_i(u) & \text{if } g(i,u) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Bellman's optimality equation and the optimal relative value function Let $J^*(i)$ denote the optimal average reward accumulated under "steady state" operation, while starting the system at state i and executing the optimal policy. Then, by virtue of the fact that the structure of the underlying CT-MDP is *communicating*, $J^*(i) = \lambda^*$ for all states i [17], and furthermore, there exists a function $h^*(i),\ i \in \mathcal{S}_\mathcal{V}$, that satisfies the following equation, for all states $i \in \mathcal{S}_\mathcal{V}$,

$$h^*(i) = \max_{u \in U(i)} \left[G(i,u) - \lambda^* \bar{\tau}_i(u) + \sum_{j \in \mathcal{S}_\mathcal{V}} p_{ij}(u) h^*(j) \right] \tag{8}$$

Function $h^*(i)$ is known as the optimal *relative value function*[7] and it defines a deterministic stationary optimal policy π^* for the considered problem by setting

$$u^*(i) \in \arg\max_{u \in U(i)} \left[G(i,u) - \lambda^* \bar{\tau}_i(u) + \sum_{j \in \mathcal{S}_\mathcal{V}} p_{ij}(u) h^*(j) \right] \tag{9}$$

Uniformization From a computational standpoint, it is more convenient to work with a *uniformized* version of Equations 8, 9: Letting $0 < \gamma < \bar{\tau}_i(u),\ \forall i, u$, and setting $\tilde{p}_{ij}(u) = \gamma p_{ij}(u)/\bar{\tau}_i(u)$ for $i \neq j$; $\tilde{p}_{ii}(u) = 1 - \gamma/\bar{\tau}_i(u)$; $\tilde{h}^*(i) = h^*(i)/\gamma$ for all i, we get the following discretized version of Equations 8, 9: for all states $i \in \mathcal{S}_\mathcal{V}$,

$$\tilde{h}^*(i) = \max_{u \in U(i)} \left[g(i,u) - \lambda^* + \sum_{j \in \mathcal{S}_\mathcal{V}} \tilde{p}_{ij}(u) \tilde{h}^*(j) \right] \tag{10}$$

$$u^*(i) \in \arg\max_{u \in U(i)} \left[g(i,u) + \sum_{j \in \mathcal{S}_\mathcal{V}} \tilde{p}_{ij}(u) \tilde{h}^*(j) \right] \tag{11}$$

The Linear Programming Approach There are several exact solution methods to solve the resulting Discrete Time Average Reward MDP (DT-AR-MDP) problem, including Value and Policy Iteration, Linear Programming (LP) and a number of variations of these basic methods. In the rest of this work we focus on the Linear Programming approach since it will allow us (i) to provide closed-form rigorous characterizations of the proposed approximation framework, and (ii) in the context of the numerical experimentation pursued in this work, it is readily implementable through the commercially available LP solvers. According to [20], the *primal* LP formulation for the considered DT-AR-MDP problem employs the decision variables λ and $\tilde{h}(i),\ i \in \mathcal{S}_V$, and its detailed structure is as follows:

$$\min \ \lambda \tag{12}$$

s.t.

$$\lambda + \tilde{h}(i) \geq g(i,u) + \sum_{j \in \mathcal{S}_\mathcal{V}} \tilde{p}_{ij}(u) \tilde{h}(j), \qquad \forall\ i \in \mathcal{S}_\mathcal{V}, u \in U(i). \tag{13}$$

We notice that (λ^*, $\tilde{h}^*$) employed in Equation 10 is *an optimal solution* of this LP. The same is true for λ^* and $\tilde{h}^* + ce$, where c is any scalar and e is the vector with all its components equal to one. Furthermore, in any optimal solution ($\bar{\lambda}$, $\bar{h}$) of the LP of Equations 12 and 13, we have $\bar{\lambda} = \lambda^*$. However, $\bar{h}$ might fail to satisfy Bellman's equation, and therefore we need to consider the *dual* LP of the LP of Equations 12 and 13 in order to obtain an *optimal* relative value function and its corresponding policy. This dual LP is formulated as follows:

$$\max \sum_{i \in \mathcal{S}_\mathcal{V}} \sum_{u \in U(i)} g(i,u) x(i,u) \tag{14}$$

s.t.
$\forall\ i \in \mathcal{S}_\mathcal{V}$,

$$\sum_{u \in U(i)} x(i,u) - \sum_{j \in \mathcal{S}_\mathcal{V}} \sum_{u \in U(j)} \tilde{p}_{ij}(u) x(j,u) = 0 \tag{15}$$

$$\sum_{i \in \mathcal{S}_\mathcal{V}} \sum_{u \in U(i)} x(i,u) = 1 \tag{16}$$

$$\forall\ i \in \mathcal{S}_\mathcal{V}, u \in U(i), \qquad x(i,u) \geq 0. \tag{17}$$

The variables $x(i, u)$ can be interpreted as the steady-state probabilities that state i will be visited and control u will then be applied. Therefore, an optimal solution x^* suggests an optimal randomized scheduling policy, where controls u at state i are selected with probability

$$\frac{x^*(i,u)}{\sum_{u \in U(i)} x^*(i,u)}. \tag{18}$$

According to [17], there will exist an optimal solution x^* with $x^*(i, u) > 0$ for only one control u at each state i with $\sum_{u \in U(i)} x^*(i, u) > 0$. The policy derived through Equation 18 from such an optimal solution, for the restricted class of states i with $\sum_{u \in U(i)} x^*(i, u) > 0$, induces a recurrent Markov chain on the considered state space. The extension of this policy to an optimal *unichain* policy for the entire set of states can be performed as follows:

Let $S_{x^*} = \{i \in \mathcal{S}_\mathcal{V} : \sum_{u \in U(i)} x^*(i, u) > 0\}$ and $\mu^*(i)$ be an optimal action at state i for which $x^*(i, \mu^*(i)) > 0$.

(i) If $\mathcal{S}_\mathcal{V} \backslash S_{x^*} = \emptyset$, stop
(ii) Find a state $s \in \mathcal{S}_\mathcal{V} \backslash S_{x^*}$ and an action $u \in U(s)$ for which $\sum_{j \in S_{x^*}} \tilde{p}_{sj}(u) > 0$
(iii) Set $S_{x^*} = S_{x^*} \cup \{s\}$ and $\mu^*(s) = u$. Go to (i)

[7]Function $h^*(\cdot)$ can be interpreted as the *asymptotic relative difference in total reward* that results from starting the CRL at the various states i and subsequently operating it according to an optimal scheduling policy; i.e., the difference $h^*(i) - h^*(j)$ expresses the asymptotic difference in total reward that results from starting the CRL in state i instead of state j, and subsequently operating it according to an optimal scheduling policy [17].

The optimal relative value function, $\tilde{h}^*(i)$, for this optimal policy can be computed by solving the following system of linear equations:[8]

$$\lambda + \tilde{h}(i) = g(i, \mu^*(i)) + \sum_{j \in \mathcal{S}_{\mathcal{V}}} \tilde{p}_{ij}(\mu^*(i))\tilde{h}(j), \qquad \forall\, i \in \mathcal{S}_{\mathcal{V}}. \tag{19}$$

Example The dual LP formulation for the DT-AR-MDP problem for the CRL of Figure 1 is as follows:

$$\max\ \mu_3\left[x(2,u_{2,2})+x(5,u_{5,1})+x(6,u_{6,1})+x(7,u_{7,1})\right] \tag{20}$$

s.t.

$$\begin{aligned}
\gamma\mu_1 x(0,u_{0,1}) - \gamma\mu_3 x(2,u_{2,2}) &= 0\\
\gamma(\mu_1+\mu_2)x(1,u_{1,1}) + \gamma\mu_2 x(1,u_{1,2}) - \gamma\mu_1 x(1,u_{0,1}) &= 0\\
\gamma\mu_1 x(2,u_{2,1}) + \gamma\mu_3 x(2,u_{2,2}) - \gamma\mu_2 x(1,u_{1,2})&\\
-\gamma\mu_3 x(7,u_{7,1}) - \gamma\mu_2 x(8,u_{8,1}) &= 0\\
\gamma\mu_2 x(3,u_{3,1}) - \gamma\mu_1 x(1,u_{1,1}) - \gamma\mu_1 x(8,u_{8,2}) &= 0\\
\gamma\mu_1 x(4,u_{4,1}) - \gamma\mu_2 x(1,u_{1,1}) - \gamma\mu_2 x(8,u_{8,2}) &= 0\\
\gamma(\mu_2+\mu_3)x(5,u_{5,1}) - \gamma\mu_1 x(2,u_{2,1}) - \gamma\mu_1 x(4,u_{4,1}) &= 0\\
\gamma(\mu_2+\mu_3)x(6,u_{6,1}) - \gamma\mu_2 x(3,u_{3,1}) &= 0\\
\gamma\mu_3 x(7,u_{7,1}) - \gamma\mu_2 x(5,u_{5,1}) - \gamma\mu_2 x(6,u_{6,1}) &= 0\\
\gamma\mu_2 x(8,u_{8,1}) + \gamma(\mu_1+\mu_2)x(8,u_{8,2})&\\
-\gamma\mu_3 x(5,u_{5,1}) - \gamma\mu_3 x(6,u_{6,1}) &= 0\\
x(0,u_{0,1}) + x(1,u_{1,1}) + x(1,u_{1,2}) + x(2,u_{2,1})&\\
+x(2,u_{2,2}) + x(3,u_{3,1}) + x(4,u_{4,1}) + x(5,u_{5,1})&\\
+x(6,u_{6,1}) + x(7,u_{7,1}) + x(8,u_{8,1}) + x(8,u_{8,2}) &= 1\\
\forall\, i \in \mathcal{S}_{\mathcal{V}}, u \in U(i), \quad x(i,u) &\ge 0
\end{aligned} \tag{21}$$

This LP can be solved optimally for any assignment of parameter values for μ_i, $i = 1, 2, 3$, and γ, and the corresponding optimal control policy can be obtained using the procedure described above. As a complete example, if μ_i = 1 for all i, and we set γ = 0.25, an optimal objective value of the LP is 0.4444 and the corresponding optimal control policy is:

$$\mu(VS_i) = \begin{cases} u_{1,1} \text{ or } u_{1,2} & \text{for } i = 1\\ u_{2,1} & \text{for } i = 2\\ u_{8,2} & \text{for } i = 8\\ u_{i,1} & \text{otherwise,}\end{cases} \tag{22}$$

where $u_{i,k}$ is the k-th control associated with state VS_i in Figure 2. In the optimal control policy, state VS_1 has two alternative optimal controls u_{11} and u_{12}, corresponding to deliberately idling the server in workstation W_1 or not, and resulting in the same optimal communicating class, for which state VS_1 is a transient state. It is interesting to notice that the actual optimal throughput is strictly less than the *nominal bottleneck throughput* of 0.5, defined by the bottleneck workstation W_1, an effect that results from the additional idleness experienced by the server due to the finite buffering capacity. $\triangle$

[8]This system of linear equations defines $\tilde{h}^*(\cdot)$ only up to translation.

The MPD-based modeling framework developed in this section provides an analytical basis for addressing the CRL scheduling problem and it can be used for the computation of the optimal scheduling policy in small CRL configurations. However, the approach has a severe computational limitation in that it requires the explicit enumeration of the underlying state space, which explodes very fast. We remind the reader that, according to the introductory discussion, it is further believed that the deployment of the CRL optimal scheduling policy is a computationally intractable problem. Hence, there is a need for some near-optimal approximating scheme to it that maintains computational tractability. This is the topic of the next two sections.

III. A Neuro-Dynamic Programming Approach for the Development of an Approximating Scheduling Policy

NDP-based approximation of relative value function through linear approximating architectures The observation that the optimal control policy for a DT-AR-MDP problem is a "greedy" policy with respect to the optimal relative value function $\tilde{h}^*$ suggests that one potential approach to generate a polynomial approximating solution to the considered scheduling problem is through the approximation of the optimal relative value function with an appropriately parameterized function. As it was mentioned in the introductory discussion, in the context of the emerging theory of *Neuro-Dynamic Programming* (NDP) [13], this parameterized function is known as the employed *approximation architecture*, and it defines the space/set of functions to be considered as candidates for the approximation. In the following, we consider a particular class of approximation architectures that are known as *linear*. Structurally, they constitute a *weighted sum* of some preselected *feature functions*, that capture important aspects of the system state and their selection is driven by practical experience, insight and/or any formal results available for the considered problem. The main intention of the work presented in this paper is to investigate the ability of the aforementioned linear architectures to provide effective approximations for the optimal relative value function of the MDP problem formulated in the previous section, while identifying a *"good"* set of feature functions to be employed by them.

A more formal characterization of feature-based parametric representational methods is as follows: Let Φ be a *feature space*, i.e., an (ordered) set $\Phi = (\phi_0, \ldots, \phi_K)$ of functions polynomially evaluated on any given state $i \in \mathcal{S}_{\mathcal{V}}$, with $\phi_j = (\phi_j(0), \ldots, \phi_j(|\mathcal{S}_{\mathcal{V}}|-1))^T, j = 0, \ldots, K$; $|\Phi| = K+1$; and $\phi_0(i) = 1$ for all $i \in \mathcal{S}_{\mathcal{V}}$. Then, any other function $f()$ defined on $\mathcal{S}_{\mathcal{V}}$ can be potentially approximated through a linear combination of the feature space Φ, by pertinently selecting a vector of *weighting coefficients* $r = (r_0, \ldots, r_K)^T$. In the application context of the considered CRL scheduling problem, we are especially interested in developing such an approximating architecture for the optimal relative value function $\tilde{h}^*()$:

$$\hat{h}(i, r) = \sum_{k=0}^{K} \phi_k(i) r_k = (\Phi r)(i). \tag{23}$$

A weight set r^* satisfying $\tilde{h}^*(i) = \hat{h}(i, r^*) = (\Phi r^*)(i),\ \forall i$, would give us the optimal relative value function and the corresponding *"greedy"* policy based on Equation 11 would be optimal. We notice, however, that it is not easy to find such a rich set Φ, while maintaining computational tractability, and as a compromising objective, we set out to find Φ and r^* such that $\tilde{h}^*(i) \approx \hat{h}(i, r^*) = (\Phi r^*)(i)$, in the sense that (i) they minimize some distance metric characterizing the quality of the approximation, and (ii) the corresponding "greedy" policy defined by Equation 11 tends to maximize throughput for the underlying DT-AR-MDP problem. Next, we elaborate on each of these two issues.

Approximating optimal relative value functions using a feature-based max-norm projection The quality – or "*goodness-of-fit*" – of the aforementioned approximation of $\tilde{h}^*()$ by $\hat{h}(i, r^*)$ can be measured using a number of distance metrics. In this paper, we consider the l_∞-norm, defined as follows:

$$\max_i |\tilde{h}^*(i) - \hat{h}(i, r^*)| \tag{24}$$

By employing the l_∞-norm in the selection of r^*, we are essentially trying to bound the distance between the optimal relative value function and its approximated value as uniformly as possible over all states. Then, a key question that is implicitly raised in this part of work is the extent to which a small uniform approximation error preserves the shape of the optimal relative value function under consideration. The detailed mathematical formulation of the weight selection problem for any feature space Φ and optimal relative value function $\tilde{h}^*(i)$, is as follows:

$$\min_{r,\epsilon}\ \epsilon \tag{25}$$

s.t.

$$|\tilde{h}^*(i) - \hat{h}(i, r)| \le \epsilon, \qquad \forall\ i \in \mathcal{S}_\mathcal{V} \tag{26}$$

This formulation can be easily transformed into an LP and solved by some LP-solving method. Notice that the dependency of the l_∞-norm on r is piecewise linear, which might lead to the existence of alternative optimal solutions for r^*.

Computing the throughput obtained by the "greedy" policy defined by $\hat{h}(i, r^*)$ The throughput of the "greedy" policy defined by $\hat{h}(i, r^*)$ can be evaluated through standard techniques provided by the MDP theory [17]. We notice, however, that this policy might present *multi-chain* structure, and this issue must be explicitly addressed by the applied algorithms. Next, we deal with this issue in the broader context of some additional practical considerations.

Some further practical considerations From a more practical standpoint, we shall eventually assess the performance of the proposed approximating scheme by comparing the throughput of the policy generated by the approximation, with the optimal throughput, λ^*, and also, the throughput that would be obtained if some other heuristics were applied. However, the implementation of this evaluation scheme is complicated and, to some extent, compromised by the following issues:

- Some of the involved computations present numerical instability and the accrued errors should be filtered out to the extent possible.
- The Mathematical Programming (MP) formulation of Equations 25 and 26 might have alternative optimal solutions r^*, resulting in different policies with different throughput. However, it is not practically possible to generate all alternative optimal solutions r^* and systematically compare their performance.
- Even worse, there might exist alternative optimal solutions $\tilde{h}_1^*$ and $\tilde{h}_2^*$ to Equation 10, with $\tilde{h}_1^* \neq \tilde{h}_2^* + ce$; such solutions can result in alternative parameterizations of the MP formulation of Equations 25 and 26, and additional approximations of the optimal scheduling policy.

One way to reduce the effects of those undesired biases is by opting to consider a broader set of actions in the determination of the final control policy, rather than only those selected by the strictly "greedy" scheme of Equation 11. In addition, we recognize that in the eventual implementation of the proposed approximating framework, the adopted policy will be the converging outcome of a *"learning"* process that will tune the weights r while employing a randomizing mechanism in the underlying decision-making process;[9] the impact of this randomization effect should also be accounted for when assessing the performance resulting from the proposed approach. On the positive side, this randomizing effect restores the "unichain"[10] structure of the considered policy. To capture all the effects discussed above, we propose to assess the performance of the considered approximating scheme through a randomizing policy that employs two different action-selection probabilities at each decision epoch: In particular, the control actions in $U(i)$ for each state i, are classified to those in $\widetilde{U}(i)$ that present considerably high value, based on $\hat{h}(i, r^*)$, and those in $U(i) \backslash \widetilde{U}(i)$. Control actions from $\widetilde{U}(i)$ are selected uniformly with some cumulative probability w, and similarly actions in $U(i) \backslash \widetilde{U}(i)$ are selected uniformly with cumulative probability $1 - w$; typically, $w \to 1$. The detailed mathematical characterization for these ideas, and the mathematical programming formulation computing the throughput of the resulting policy are as follows:

$$\forall\ i \in \mathcal{S}_\mathcal{V},\ u \in U(i), \quad R(i, u) = g(i, u) + \sum_{j \in \mathcal{S}_\mathcal{V}} \tilde{p}_{ij}(u) \hat{h}(j, r^*) \tag{27}$$

$$\begin{aligned} \forall\ i \in \mathcal{S}_\mathcal{V},\ \widetilde{U}(i) \quad &= \quad \{u : \max_{u \in U(i)} R(i, u) - R(i, u) \le \\ &\qquad \delta\, \big|\max_{u \in U(i)} R(i, u)\big|, u \in U(i)\} \end{aligned} \tag{28}$$

$$\min\ \lambda_R(r^*; \delta, w) \tag{29}$$

s.t.

$\forall\ i \in \mathcal{S}_\mathcal{V}, \quad \lambda_R(r^*; \delta, w) + \tilde{h}(i)$

[9]This randomizing mechanism essentially compensates for the fact that any approximating policy derived from the observation and analysis of a limited number of sample paths of the system behavior, is based on imperfect information about the system dynamics and the accompanying value function, and therefore, it cannot be fully trustworthy.

[10]A *unichain* policy is a stationary policy that confines the system operation on a subspace corresponding to a Markov chain with a single recurrent class and a possibly empty set of transient states.

$$= \begin{cases} \frac{w}{\left|\widetilde{U}(i)\right|} \sum_{u \in \widetilde{U}(i)} \left[g(i,u) + \sum_{j \in \mathcal{S}_{\mathcal{V}}} \tilde{p}_{ij}(u) \tilde{h}(j) \right] \\ + \frac{1-w}{\left|U(i)\right| - \left|\widetilde{U}(i)\right|} \sum_{u \in U(i) \setminus \widetilde{U}(i)} \Big[g(i,u) + \\ \sum_{j \in \mathcal{S}_{\mathcal{V}}} \tilde{p}_{ij}(u) \tilde{h}(j) \Big], \quad \text{if } \left|U(i)\right| \neq \left|\widetilde{U}(i)\right| \\ \frac{1}{\left|\widetilde{U}(i)\right|} \sum_{u \in \widetilde{U}(i)} \left[g(i,u) + \sum_{j \in \mathcal{S}_{\mathcal{V}}} \tilde{p}_{ij}(u) \tilde{h}(j) \right], \\ \qquad\qquad\qquad \text{if } \left|U(i)\right| = \left|\widetilde{U}(i)\right| \end{cases} \tag{30}$$

The parameter δ appearing in Equation 28 controls the degree of "greediness" of the resulting policy; typically it should take positive values close to 0. Having detailed the mathematical apparatus that is necessary for the performance evaluation of the proposed approximating scheme, in the next section we consider the selection of a particular set of features that could lead to good approximations of the optimal CRL scheduling policy.

IV. Selecting a Feature Space Φ for the CRL Scheduling Problem: An Experimental Investigation

A. Suggesting a Feature Space Φ

Extracting feature functions Identifying feature functions is a kind of data compression process that seeks to incorporate application-specific domain knowledge into the data representation. Therefore, it is very application driven, in general. In our case, the feature selection process is based on a number of queueing-theoretic concepts and results [21], [22], and it will seek to capture the following information:

- Basic State Information
  - number of jobs waiting, in processing, or being finished at each job stage.
  - existence of job instances waiting, in processing, or being finished at each job stage.
  - buffer occupancy / availability at each workstation.
- Interactions
  - Interactions between the feature elements characterizing the basic state information

We notice that in [22], [21], similar information was employed for predicting performance bounds of queueing networks modeling re-entrant lines. Furthermore, the work of [23] constructed an approximation function of degree 2 or 3 using basic functions representing the number of jobs at each stage, and showed that good fits to the optimal value function were possible for several types of uncapacitated queueing networks.

A detailed characterization of the feature functions employed in this work, seeking to capture the basic state information listed above, is provided in Table III. We shall refer to this set of features as *simple features*, since they can be computed directly as simple functions of the system state vector. Interactions of simple features are captured by a set of *composite features* that essentially constitute pairwise products of simple features.[11] Finally, we group feature functions into *"classes"*, with each class containing all the feature functions resulting by the application of the same feature concept on different components of the underlying CRL.

TABLE III

Simple Features

Class	Expression
SF0	1
SF1	$n_{j,w}, j = 2, \ldots, M$
SF2	$n_{j,p}, j = 1, \ldots, M$
SF3	$n_{j,b}, j = 1, \ldots, M-1$
SF4	$I_{\{n_{j,w}>0\}}, j = 2, \ldots, M$
SF5	$I_{\{n_{j,p}>0\}}, j = 1, \ldots, M$
SF6	$I_{\{n_{j,b}>0\}}, j = 1, \ldots, M-1$
SF7	$I_{\{\sum_{j \in \sigma(W_i)} (n_{jw}+n_{jp}+n_{jb})=0\}}, i = 1, \ldots, L$
SF8	$I_{\{0 \le \sum_{j \in \sigma(W_i)} (n_{jw}+n_{jp}+n_{jb}) \le 0.2B_i\}}, i = 1, \ldots, L$
SF9	$I_{\{0.2B_i \le \sum_{j \in \sigma(W_i)} (n_{jw}+n_{jp}+n_{jb}) \le 0.8B_i\}}, i = 1, \ldots, L$
SF10	$I_{\{0.8B_i \le \sum_{j \in \sigma(W_i)} (n_{jw}+n_{jp}+n_{jb}) \le B_i\}}, i = 1, \ldots, L$
SF11	$I_{\{\sum_{j \in \sigma(W_i)} (n_{jw}+n_{jp}+n_{jb})=B_i\}}, i = 1, \ldots, L$
SF12	$B_i - \sum_{j \in \sigma(W_i)} (n_{jw} + n_{jp} + n_{jb}), i = 1, \ldots, L$

Complexity of the suggested set Φ of feature functions The above feature specification results in 91 classes, including a total of $M(18M + 36L - 22) + L(18L - 35) + 7$ feature functions.[12] While it is true that, in general, we can increase the representational capability of a feature space Φ by adding more composite features, such an expansion will also increase the computational complexity of the approximation. Hence, in the first part of our work, we suggest a minimalist approach, restricting the degree of employed composite features to 2; the impact of adding more composite features, corresponding to higher-order interactions of simple features, is addressed in Section IV-D.

Using these feature functions, a linearly parameterized approximation function as defined in Equation 23 is established, and the parameter vector, r, is computed based on the l_∞-norm projection of the optimal relative value function to the corresponding feature space Φ, using Equations 25 and 26. The evaluation of the approximating capability of feature space Φ was performed through the following numerical experiment.

B. A Numerical Experiment for Evaluating Φ

We tested the potential performance of the approximating architecture generated by the aforementioned feature functions, on two types of re-entrant line, the first consisting of 2 *single-server* workstations and the second consisting of 3 *single-server* workstations. Both of these lines are observing the operational assumptions stated in the previous sections, while the adopted SCP was the *optimal* – i.e., *maximally permissive* – policy. For each type of re-entrant line, different configurations were generated by changing buffering capacities; Table IV summarizes the system configurations used in this experiment. For each configuration, 30 problem instances with randomly generated processing rates were considered.

[11]However, we omit products that result to feature functions that are identical to one of its constituent factors.

[12]We notice that some important information such as (*immediate* or *total*) workload of a workstation, that is typically considered by queueing theory, is not considered explicitly in our feature specification since it can be represented by a linear combination of the employed feature functions.

TABLE IV

SYSTEM CONFIGURATIONS CONSIDERED IN THE NUMERICAL EXPERIMENT

configurations	number of workstations	number of job stages (JS) and job routes	buffer capacities
Conf 1	2	3JS($W_1 \to W_2 \to W_1$)	(B_1, B_2)=(1,2)
Conf 2			(B_1, B_2)=(3,2)
Conf 3			(B_1, B_2)=(4,4)
Conf 4	3	4JS($W_1 \to W_2 \to W_3 \to W_1$)	(B_1, B_2, B_3)=(1,2,2)
Conf 5			(B_1, B_2, B_3)=(3,2,2)
Conf 6			(B_1, B_2, B_3)=(4,3,2)

TABLE V

THE NUMBER OF STATES AND FEATURE FUNCTIONS FOR THE CONSIDERED RE-ENTRANT LINES

Configurations	number of states	number of feature functions
Conf 1	9	255
Conf 2	70	
Conf 3	275	
Conf 4	85	563
Conf 5	460	
Conf 6	1079	

The number of states generated in each case, and the number of the employed feature functions, are summarized in Table V.

Experimental results and assessment To assess the performance of the considered architecture in each case, we computed the throughput λ^* resulting from the optimal policy, and also the throughput that would be attained by the randomized policy defined by the approximating relative value function, $\hat{h}(i, r^*)$, according to the logic outlined in Section III. More specifically, the throughput of the randomized policy was computed while increasing the value δ from 0 to 0.020 by 0.001, and the value w from 0.80 to 0.99 by 0.01. We define the % error for this policy by

$$\%error = \frac{Optimal\ TH - TH\ by\ rand.\ policy}{Optimal\ TH} \times 100. \tag{31}$$

We also compared the % error attained by the proposed architecture to the % error generated by some known heuristics that have been shown to perform well in the case of uncapacitated re-entrant lines, namely, the *Last Buffer First Serve (LBFS), First Buffer First Serve (FBFS), First In First Out (FIFO)*, and *Least Work Next Queue (LWNQ)* policies. Tables VI and VII summarize the obtained results. More specifically, Table VI lists the average, minimum and maximum % errors of throughput obtained by using the aforementioned heuristics on each of the six CRL configurations, while Table VII reports the results characterizing the performance of the randomized policy obtained through the method of Section III. Columns 2 and 3 in Table VII report the values of the parameters (δ, w) that resulted in the best performance for the generated policy. Column 4 reports the average of the l_∞-norm approximation errors characterizing the goodness-of-fit for each of the 30 problem instances generated for each configuration. Columns 5, 6 and 7 show respectively the average, minimum, and maximum % errors achieved by the proposed approximating method when using the feature space Φ detailed above. Finally, Column 8 provides a measure of the "non-greediness" of the derived policy, by reporting the extra number of control actions included in $\tilde{U}(i)$, averaged over all states i.

Some interesting remarks regarding the results of this numerical experiment and their implications for the quality of the proposed approximating method, can be summarized as follows:

TABLE VI

PERFORMANCE OF HEURISTICS FOR THE CONSIDERED RE-ENTRANT LINES

Config.	% error	FBFS capacity	LBFS	FIFO	LWNQ-FBFS	LWNQ-LBFS	LWNQ-FIFO
Conf 1	Avg.	0	2.906683	0	0	0	0
	Min.	0	1.419307	0	0	0	0
	Max.	0	5.651860	0	0	0	0
Conf 2	Avg.	2.088194	2.865571	2.088194	1.889512	2.427890	1.889512
	Min.	0.002057	0.000411	0.002057	0.002057	0.000411	0.000206
	Max.	6.013643	10.944309	6.013643	7.432376	8.647182	7.432376
Conf 3	Avg.	0.593519	1.322035	0.593519	0.625181	0.671251	0.625181
	Min.	0	0	0	0	0	0
	Max.	4.298884	9.824424	4.298884	4.964331	5.110277	4.964331
Conf 4	Avg.	0.802350	2.657095	0.802350	0.802350	0.802350	0.802350
	Min.	0	0.308027	0	0	0	0
	Max.	2.831750	14.249711	2.831750	2.831750	2.831750	2.831750
Conf 5	Avg.	3.043938	4.621490	3.043938	2.534003	2.934190	2.534003
	Min.	0.095917	0.082142	0.095917	0.085206	0.086738	0.085206
	Max.	8.457638	14.164897	8.457638	6.956313	7.461679	6.956313
Conf 6	Avg.	2.783108	1.751215	2.783108	0.988989	1.051021	0.988989
	Min.	0.099751	0.000172	0.099751	0.000172	0.000172	0.000172
	Max.	8.869749	5.477629	8.869749	4.231201	4.535971	4.231201

- Overall, the throughput errors generated by the proposed approach are rather small.
- Furthermore, the randomized policy derived with the selected values (δ, w), has lower average % errors than the errors attained by the considered heuristics. In fact, it was found that this dominance is quite robust with respect to the exact values of δ, w. A sensitivity analysis of the randomized policy with respect to the parameter vector (δ, w) indicated that the value of w should be kept very close to one, while δ should maintain low values, maybe in the range of [0, 0.01].
- Even more importantly, the randomized policy is more consistent in its performance than the considered heuristics, as manifested by the reported maximum % errors.
- It is also interesting to notice that for the case of Configuration 1, the reported throughput error is non-zero, even though the employed architecture supports perfect goodness-of-fit. This results from the randomizing nature of the derived policy.
- The reported non-zero value for ϵ^* for the cases of Configurations 2, 4 and 5, when combined with the data of Table V, imply that the rank of the feature matrix Φ must be quite small, i.e., there must be considerable linear dependency among the employed features. We believe that this problem will be alleviated for CRL's with larger buffering capacities, since in that case there will be more differentiation among the values of the various simple features.

C. Consideration of Scalability

The experiment reported in Section IV-B employed quite small system configurations, in an effort to maintain compu-

TABLE VII

PERFORMANCE OF THE RANDOMIZED POLICY GENERATED BY THE PROPOSED ARCHITECTURE USING Φ

Config.	δ	w	Avg. ϵ^*	Avg. % error	Min. % error	Max. % error	Avg. # of additional controls per state
Conf 1	0.000	0.99	0	0.093051	0.016866	0.173766	0
Conf 2	0.001	0.98	0.934340	0.820087	0.003095	4.886354	0.079524
Conf 3	0.011	0.99	1.578569	0.714999	0.000104	3.523133	0.634667
Conf 4	0.004	0.99	0.828601	0.525297	0.065391	1.908183	0.025490
Conf 5	0.004	0.99	1.977419	0.723601	0.003318	2.617387	0.144638
Conf 6	0.007	0.99	2.917650	0.727640	0.000625	3.502581	0.303182

TABLE VIII

CONSIDERED SYSTEM CONFIGURATIONS FOR THE SCALABILITY TEST

Config.	number of workstations	number of job stages (JS) and job routes	buffering capacities
Conf 7	2	3JS($W_1 \rightarrow W_2 \rightarrow W_1$)	(B_1, B_2)=(10,10)
Conf 8	3	4JS($W_1 \rightarrow W_2 \rightarrow W_3 \rightarrow W_1$)	(B_1, B_2, B_3)=(5,5,6)
Conf 9	4	7JS($W_1 \rightarrow W_2 \rightarrow W_4$ $\rightarrow W_1 \rightarrow W_2 \rightarrow W_3 \rightarrow W_1$)	(B_1, B_2, B_3, B_4)=(3,2,1,2)

tational tractability. In this section, we report some additional experiments indicating that the previously found results pertain also to larger system configurations, where the number of the system states is significantly greater than the number of the employed feature functions.

Design of a numerical experiment for larger-sized systems We can generate larger-sized re-entrant lines by increasing the number of workstations, the number of job stages, or the buffering capacity. However, these elements also increase drastically the number of the system states, to the extent that the solution of the MP formulations of Equations 14 – 17, Equations 25 – 26, and Equations 29 – 30, that are employed in this work for implementing the proposed approximating scheme, becomes very cumbersome. Our intention in this experiment is to generate large systems that we can still handle, in the sense that the MP formulations of Equations 14 – 17, Equations 25 – 26, and Equations 29 – 30 can be solved in reasonable time. Hence, we considered two types of "large-sized" re-entrant lines, the first generated by increasing the buffering capacity of the re-entrant lines considered in Section IV-B, and the second generated by increasing the number of workstations and job stages.[13] The first type of the proposed expansion has the interesting effect that it generates a large number of states while maintaining a small number of feature functions, and therefore, it allows us to test the data-compressing capability of the considered set Φ of feature functions. The second type of the proposed expansion intends to assess the scalability of the previously obtained results in the face of more complex process flows. Tables VIII and IX summarize the considered configurations and quote the number of states and the feature functions generated by the approximating procedure. Configurations 7 and 8 are expansions of Configurations 3 and 6, resulting from increasing buffering capacities to $B = (10, 10)$ and $B = (5, 5, 6)$, respectively. Configuration 9 is a re-entrant line consisting of 4 single-server workstations with buffering capacities $B = (3, 2, 1, 2)$, and a process route of 7 job stages, depicted in Figure 3. For all these configurations, a numerical experiment was performed by using (i) the set Φ of feature functions including up to 2-order interactions and

[13]Increasing all these elements at the same time generates a huge number of states and makes the considered scheduling problem computationally intractable.

TABLE IX

THE NUMBER OF STATES AND FEATURE FUNCTIONS IN THE CONSIDERED SYSTEM CONFIGURATIONS FOR THE SCALABILITY TEST

Config.	number of states	number of feature functions
Conf 7	3872	255
Conf 8	10018	563
Conf 9	21093	1497

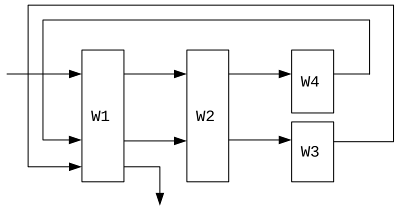

Fig. 3. A re-entrant line with 4 single-server workstations and 7 job stages

(ii) 10 problem instances with randomly generated processing rates.

A numerical experiment for Configurations 7 and 8 Tables X and XI report respectively the results characterizing the performance of the randomized policy obtained through the method in Section III and the performance of some heuristics, when applied on Configurations 7 and 8. These results are consistent with the ones obtained in the previous subsection for smaller-sized re-entrant lines, in the sense that the randomized policy has lower average and maximum % errors compared to the errors attained by the considered heuristics; these results are also robust with respect to the values of the parameter vector (δ, w). From a more qualitative standpoint, they indicate that the considered set Φ of feature functions has a good representational capability even for the case that the number of states is quite greater than the number of feature functions. Finally, notice that the average values of the optimal approximation error, ϵ^*, are increased compared to those in Section IV-B. This increase results from the fact that, even though the linear independency among the employed feature functions is enhanced by increasing buffering capacities, the number of states is increased significantly more, so that the rank of the feature matrix Φ becomes quite smaller than the number of states. Interestingly, this deterioration of the "goodness-of-fit" of the relative value function does not seem to affect the performance of the resulting "greedy" policies.

TABLE X

PERFORMANCE OF THE RANDOMIZED POLICY GENERATED BY THE PROPOSED ARCHITECTURE FOR CONFIGURATIONS 7 AND 8

Config.	δ	w	Avg. ϵ^*	Avg. % error	Min. % error	Max. % error	Avg. # of additional controls per state
Conf 7	0.000	0.99	12.353925	0.116272	0.000603	0.550340	0.000646
Conf 8	0.001	0.98	6.264441	0.074681	0.000395	0.183286	0.061409

TABLE XI

PERFORMANCE OF HEURISTICS FOR CONFIGURATIONS 7 AND 8

Config.	% error	FBFS	LBFS	FIFO	LWNQ-FBFS	LWNQ-LBFS	LWNQ-FIFO
Conf 7	Avg.	0.550538	1.053770	0.550538	0.537978	0.563585	0.537978
	Min.	0	0	0	0	0	0
	Max.	2.823672	7.421845	2.823672	2.753983	2.978492	2.753983
Conf 8	Avg.	0.546995	0.232962	0.546995	0.148092	0.191303	0.148092
	Min.	0.000130	0.000780	0.000130	0	0	0
	Max.	2.755537	0.946889	2.755537	0.705782	0.948396	0.705782

TABLE XII

PERFORMANCE OF THE RANDOMIZED POLICY GENERATED BY THE PROPOSED ARCHITECTURE WITH w = 0.99 FOR CONFIGURATION 9

Config.	δ	Avg. ϵ^*	Avg. % error	Min. % error	Max. % error	Avg. # of additional controls per state
Conf 9	0.001	13.823142	3.549228	0.197071	7.473375	0.021690

TABLE XIII

PERFORMANCE OF HEURISTICS FOR CONFIGURATION 9

Config.	% error	FBFS	LBFS	FIFO	LWNQ-FBFS	LWNQ-LBFS	LWNQ-FIFO
Conf 9	Avg.	5.066853	4.306452	6.470393	4.050331	3.642792	4.114147
	Min.	0.946650	0.102316	0.684539	0.325586	0.079637	0.325977
	Max.	8.143624	11.560304	9.472767	8.691876	9.024313	8.699038

TABLE XIV

THE NUMBER OF STATES AND FEATURE FUNCTIONS IN $\Phi^{'}$ FOR THE CONSIDERED RE-ENTRANT LINES

Config.	number of states	number of feature functions in $\Phi^{'}$
Conf 1	9	1642
Conf 2	70	
Conf 3	275	
Conf 4	85	5623
Conf 5	460	
Conf 6	1079	

A numerical experiment for Configuration 9 Tables XII and XIII summarize the experimental results obtained with respect to Configuration 9. Due to the very long computational times involved in this particular experiment, we generated only 4 problem instances with randomly selected processing rates and we fixed the value of parameter w to $w = 0.99$,[14] in order to characterize the performance of the considered randomized policy. Clearly, the obtained results are consistent with those obtained for Configurations 7 and 8, in the sense that the randomized policy has lower average and maximum % errors compared to those attained by the considered heuristics.

D. Investigating the impact of adding higher-order interactions in Φ on the performance of the employed approximating scheme

Extending the Feature Space Φ by including up to 3-order Interactions This subsection investigates the impact of adding 3-order interactions in Φ on the performance of the employed approximating scheme. A new extended set $\Phi^{'}$ of feature functions considering up to 3-order interactions was generated by adding to the set Φ composite features represented by the products of three simple features. After the omission of some generated feature functions that are identical to one of the feature functions in Φ, the resulting feature set $\Phi^{'}$ consists of $M(36M^2-78M+6L+60)+ML(108M+108L-216)+L(36L^2-138L+130)-15$ feature functions, organized in 455 classes. In the provided formula, M denotes the number of job stages and L denotes the number of workstations.

As in Section IV-B, a linearly parameterized approximation function was established using these feature functions, and the quality of the approximations provided by the resulting architecture was evaluated through the following numerical experiment.

Experimental Results and Assessment For this numerical experiment, we used the same configurations and the same processing rates for each generated problem instance, that were used in Section IV-B for the evaluation of the feature set Φ. The number of the generated feature functions is summarized in Table XIV. As in Section IV-B, the actual number of classes is reduced to 364 by the fact that feature classes SF2 and SF5 are the same and therefore, only one of them should be considered; collectively, they include $\frac{M}{6}(125M^2-306M+283)+\frac{ML}{6}(450M+540L-1014)+L(36L^2-138L+130)-15$ feature functions. The evaluation of the approximating capability of the new architecture was performed according to the logic described in Section III, while considering the same values of (δ, w) as in Section IV-B, and the obtained results are presented in Table XV. Some interesting remarks regarding these results can be summarized as follows:

- The performance achieved by using the new set $\Phi^{'}$ of feature functions was improved, as indicated by the lower average values of ϵ^*, % errors, and maximum % errors, compared to those attained by the set Φ.
- In fact, a *paired t-test* [24] applied on the relevant data,

[14]We remind the reader that in the previous experimental results, the considered randomized policy has better performance when the value of w is kept very close to one.

TABLE XV

PERFORMANCE OF THE RANDOMIZED POLICY GENERATED BY THE PROPOSED ARCHITECTURE USING Φ'

Config.	δ	w	Avg. ϵ^*	Avg. % error	Min. % error	Max. % error	Avg. # of additional controls per state
Conf 1	0.000	0.99	0	0.093051	0.016866	0.173766	0
Conf 2	0.000	0.99	0	0.067502	0.000828	0.215701	0.412857
Conf 3	0.000	0.98	0.623345	0.420034	0.000305	3.184219	0.042424
Conf 4	0.000	0.99	0	0.046410	0.001023	0.193298	0
Conf 5	0.002	0.99	0.450591	0.236877	0.005249	0.692937	0.130290
Conf 6	0.003	0.99	1.096907	0.491644	0.002053	3.402330	0.156256

indicated that this performance improvement is significant with a confidence level higher than 99.95 %.[15]

- The performance of the considered randomized policy remains more consistent compared to the performance demonstrated by the applied heuristics.
- The above results are quite robust with respect to the exact values of δ and w.
- The inclusion of higher-order interactions introduces new linearly independent feature functions, a fact manifested by the smaller average values of ϵ^* in all configurations.

V. CONCLUSIONS

This paper proposed an NDP-based approximating architecture for the relative value function underlying the CT-MDP formulation of the capacitated re-entrant line scheduling problem, and assessed its performance through a numerical experiment. The derived results indicate that the proposed feature space and the induced approximating architecture hold considerable promise for providing a compact representation of the target function. Our future work intends to further validate this assumption and promote the further development of the proposed approximation framework, by systematically investigating the following research topics: (i) the development of a weight tuning algorithm that is computationally tractable for large-scale CRL configurations, (ii) the investigation of the quality of the approximations obtained through the employment of distance metrics other than the l_∞-norm, (iii) the statistical assessment of the *significance* of the various feature functions for the performance of the resulting approximation, and (iv) the extension of the developed results to production systems more general than the CRL.

ACKNOWLEDGMENT

This work was partially supported by NSF grant ECS-9979693 and by Keck Foundation.

REFERENCES

[1] P. R. Kumar, "Scheduling manufacturing systems of re-entrant lines," in *Stochastic Modeling and Analysis of Manufacturing Systems*, D. D. Yao, Ed. Springer-Verlag, 1994, pp. 325–360.

[2] J. Y. Choi and S. A. Reveliotis, "A generalized stochastic petri net model for performance analysis and control of capacitated re-entrant lines," *IEEE Trans. Robotics & Automation*, vol. 19, no. 3, pp. 474–480, 2003.

[3] S. A. Reveliotis, M. A. Lawley, and P. M. Ferreira, "Structural control of large-scale flexibly automated manufacturing systems," in *The Design of Manufacturing Systems*, C. T. Leondes, Ed. CRC Press, 2001, pp. 4–1 – 4–34.

[4] D. D. Yao Ed., *Stochastic Modeling and Analysis of Manufacturing Systems*. NY, NY: Springer-Verlag, 1994.

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

[6] P. R. Kumar, "Scheduling semiconductor manufacturing plants," *IEEE Control Systems Magazine*, vol. 14–6, pp. 33–40, 1994.

[7] S. H. Lu, D. Ramaswamy, and P. R. Kumar, "Efficient scheduling policies to reduce mean and variance of cycle-time in semiconductor manufacturing plants," *IEEE Trans. on Semiconductor Manufacturing*, vol. 7, pp. 374–385, 1994.

[8] S. Kumar and P. R. Kumar, "Fluctuation smoothing policies are stable for stochastic re-entrant lines," *Discrete-Event Dynamic Systems: Theory and Applications*, vol. 6, pp. 361–370, 1996.

[9] ——, "Queueing network models in the design and analysis of semiconductor wafer fabs," *IEEE Trans. Robotics & Automation*, vol. 17, pp. 548–561, 2001.

[10] S. A. Reveliotis, "The destabilizing effect of blocking due to finite buffering capacity in multi-class queueing networks," *IEEE Trans. on Autom. Control*, vol. 45, pp. 585–588, 2000.

[11] M. A. Lawley and S. A. Reveliotis, "Deadlock avoidance for sequential resource allocation systems: hard and easy cases," *Intl. Jrnl of FMS*, vol. 13, pp. 385–404, 2001.

[12] C. A. Papadimitriou and J. N. Tsitsiklis, "The complexity of optimal queueing network control," *Mathematics of Operations Research*, vol. 24, 1999.

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

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

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

[16] D. P. de Farias, "The linear programming approach to approximate dynamic programming: Theory and application," Ph.D. dissertation, Massachusetts Institute of Technology, June 2002.

[17] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[18] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Boston, MA: Klumwer Academic Pub., 1999.

[19] P. Glasserman and D. D. Yao, *Monotone Structure in Discrete Event Systems*. John Wiley & Sons Inc., 1994.

[20] D. P. Bertsekas, *Dynamic Programming and Optimal Control, vol. 2*. Belmont, MA: Athena Scientific, 1995.

[21] D. Bertsimas, I. C. Paschalidis, and J. N. Tsitsiklis, "Optimization of multiclass queueing networks: Polyhedral and nonlinear characterizations of achievable performance," *The Annals of Applied Probability*, vol. 4, no. 1, pp. 43–75, 1994.

[22] S. Kumar and P. R. Kumar, "Performance bounds for queueing networks and scheduling policies," *IEEE Trans. Autom. Control*, vol. 39, pp. 1600–1611, 1994.

[23] P. J. Schweitzer and A. Seidmann, "Generalized polynomial approximations in markov decision processes," *Journal of Mathematical Analysis and Applications*, vol. 110, pp. 568–582, 1985.

[24] A. J. Hayter, *Probability and Statistics*. International Thomson PUB, 1996.

[25] J. Y. Choi, "Performance modeling, analysis and control of capacitated re-entrant lines," Ph.D. dissertation, Georgia Institute of Technology, Atlanta, GA, 2004.

[15]This statistical analysis is reported in [25] and it can be obtained by contacting the authors.