Performance Modeling, Analysis and Control of Capacitated Re-entrant Lines

A Thesis
Presented to
The Academic Faculty

by

Jin Young Choi

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Industrial & Systems Engineering
Georgia Institute of Technology
July 2004

Performance Modeling, Analysis and Control of Capacitated Re-entrant Lines

Approved by:

Professor Spyros A. Reveliotis, Committee Chair	Professor Jim Dai
Professor Hayriye Ayhan	Professor Magnus Egerstedt
Professor Ron Billings	Professor Leon F. McGinnis

Date Approved: July 1, 2004

This dissertation is dedicated to

my wife, Jeong Ae

my daughter, Joo Hee and

my son, Jae Hyuk

for their sacrifice and love

ACKNOWLEDGEMENTS

First, I would like to thank God for blessing me and allowing me such opportunities as to complete my Ph.D.

I would express my sincere appreciation to my advisor, Professor Spyros A. Reveliotis for his thoughtful guidance, encouragement and support that he has generously provided throughout my Ph.D. work.

I am truly grateful to Professor Leon F. McGinnis for his academic and financial support through interesting research projects at the Keck Virtual Factory Lab.

I also thank the other committee members, Professors Hayriye Ayhan, Ron Billings, Jim Dai and Magnus Egerstedt for their time and valuable comments and suggestions on this work.

Much appreciation and love is extended to all of my family and friends. I would like to especially acknowledge my dear parents and parents-in-law, for their continued understanding, encouragement and love throughout all these years.

Most importantly, I would like to wholeheartedly thank my lovely wife, Jeong Ae and my precious daughter, Joo Hee, and son, Jae Hyuk, for their love, sacrifice, and encouragement. Thank you. I love you all.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The basic definition of the re-entrant line, which constitutes the typical abstraction for the formal modeling and analysis of the fab scheduling problem, considers only the job contest for the finite processing capacity of the system workstations, ignoring completely the effects and complications arising from additional operational issues like the finite buffering capacity of the system workstation / production units. Yet, as the semiconductor industry moves to more extensively automated operational modes, the explicit characterization and control of these additional operational features is of paramount importance for the robust and stable operation of the entire system. Moreover, the operational policies developed to control these logical aspects of the system behavior introduce additional constraints to the fab scheduling problem, that complicate it even further and, more importantly, invalidate prior characterizations of its optimal solutions.

Motivated by these remarks, this thesis considers the problem of performance modeling, analysis and control of capacitated, flexibly automated re-entrant lines. Specifically, the first part of the thesis develops an analytical framework for the modeling, analysis, and control of capacitated re-entrant lines, which is based on Generalized Stochastic Petri nets framework. The corresponding scheduling problem is systematically formulated, and the structure of the optimal control policy is characterized and compared to that identified for "traditional" re-entrant lines.

The second part of thesis addresses the problem of developing a systematic and computationally effective method for computing the optimal scheduling policy for any given configuration of capacitated re-entrant line. Specifically, the underlying scheduling problem is transformed to a Markov Decision Process (MDP) problem and an algorithm that systematically generates the MDP formulation for any given fab configuration, while leading to a substantial reduction of the underlying state space, is developed. Even though this approach is still too complex for practical implementation, it is instrumental for the design

of novel approximating schemes to the optimal scheduling policy, since it allows the experimental assessment of their quality by comparing their performance to that of the optimal policy on small system configurations.

The third part of thesis develops such an efficient approximating scheme based on the Neuro-Dynamic Programming (NDP) theory. In its general definition, the NDP method seeks the approximation of the optimal relative value function of the underlying MDP formulation by a parameterized function. Hence, an approximating structure for the considered problem is proposed and the quality of the generated approximations is systematically assessed. More specifically, this part of the thesis develops a set of "*feature*" functions and the mathematical apparatus necessary to evaluate the considered approximating scheme through a numerical experiment. The obtained results indicate that good quality approximations can be achieved by considering a set of features that characterize the distribution of the running process instances to the various processing stages and their lower (2nd or 3rd) order interactions. In particular, the resulting approximations have comparable performance to that obtained by the application of the typically used dispatching rules, but they are able to deliver this performance in a more robust and consistent manner.

The last part of the thesis exploits the performance models developed in its earlier parts in order to provide an analytical characterization of the optimality of various deadlock resolution strategies for Markovian resource allocation systems under the objective of maximizing throughput. The thesis concludes by discussing some potential extensions to this research work.

CHAPTER 1

INTRODUCTION

With the migration of modern technological applications to highly automated modes of operation, the effective and efficient deployment, reconfiguration and control of the resource allocation underlying the operation of these environments is an issue of ever-increasing importance. Yet, currently we are lacking an adequate methodology for the effective real-time management of these flexibly automated *resource allocation systems (RAS)*, partly due to the fact that past research on the performance modeling and control of these environments has adopted a high-level perspective of their dynamics, ignoring the lower-level operational details. Characteristically, it is interesting to notice that the *hierarchical decomposition* framework [16], the most widely adopted "analytical" framework for planning and control in production environments, discerns strategic, tactical, and operational decisions, all of which address performance objectives, while it presumes the logically consistent and robust system behavior. However, in an extensively automated environment, the establishment of logically correct and robust behavior to the various operational contingencies, is definitely a responsibility of the underlying control logic. This gives rise to a new set of control problems, referred to as the RAS *Structural Control (SC)* [45], which has been investigated extensively in the last decade, but the integration of the developed set of results with the complementary function of performance-oriented control remains still to be addressed.

As a case in point, consider the contemporary semiconductor manufacturing fab. Each product fabricated in this environment requires several hundreds of operations, typically taking place in a repetitive fashion, which leads to a re-entrant pattern for the underlying material flow and a very high complexity for the resulting scheduling problem. Currently, the most typical abstraction for the formal modeling and analysis of this fab scheduling problem is the *re-entrant (production) line*. In its basic characterization [23], such a line supports the production of a single item through L workstations, $W_1,\ W_2, \ldots, W_L$. Each

workstation W_i, $i = 1, \ldots, L$, possesses S_i identical servers and *infinite* buffering capacity, and the production of each unit occurs in M stages, J_1, $J_2, \ldots, J_M$; stage J_j, $j = 1 \ldots, M$, is supported by one of the system workstations, to be denoted by $W(J_j)$. The re-entrant nature of the line is expressed by the fact that there exists at least one workstation W_k such that $|\{j : W(J_j) = W_k\}| \geq 2$, and raises the problem of determining how to allocate the workstation processing capacity to the job stages competing for it, in order to optimize some pre-specified performance objective(s).[1] The resulting scheduling problem has been investigated extensively in the last decade, and many of the developed results are analytically strong and of high mathematical sophistication. A representative and insightful exposition of these results is provided in the recent survey paper of [28].

As it was mentioned above, the basic re-entrant line model considers that each workstation possesses infinite buffering capacity. This feature has been justified in the past by the presence of the human operator in the fab shop-floor, that handily addressed any potential overflow problems. The migration of modern fabs to highly automated modes of operation, through the advent of 300mm production technology, necessitates the development of explicit real-time control logic that will establish the logically correct and consistent operation of the fab shop-floor, including the orderly allocation of limited resources like the buffering capacity of the system workstations and the interconnecting material handling equipment. The corresponding set of real-time control problems is collectively known as the fab *logical* or *structural* control problem, and it is treated in [37]. As it is argued in [37], the explicit modeling in the performance control problem of these additional operational aspects and of the control policies developed to address the fab logical control problem, necessitates its systematic re-investigation. Indeed, a preliminary study of the problem of scheduling structurally controlled re-entrant lines has indicated that the introduction of the finite buffering capacity and the corresponding structural control logic into the fab operational model, leads to additional material flow dynamics, that negate in a strong qualitative sense prior analytical results, obtained through the study of the basic re-entrant line model outlined above

[1] Due to the very high capital cost of modern fabs, and the market forces that have traditionally driven the fab economics, the major performance objective addressed by the re-entrant line scheduling problem is the maximization of the system throughput.

[41].

All the prior remarks suggest that it is valuable to provide an analytical framework for addressing the fab scheduling problem in its contemporary, more complex operational context. This analytical framework must lead to enhanced operational efficiency while maintaining behavioral correctness and computational tractability, and it will constitute the starting point for the development of new scheduling tools and policies. A more detailed break-down of our research objectives is as follows:

1. An analytical framework for the modeling, analysis, and control of *capacitated re-entrant lines (CRL)* will be developed, the corresponding scheduling problem will be systematically formulated, and the structure of the optimal control policy will be characterized and compared to that identified for "traditional" re-entrant lines.

2. The results of Task 1 will be subsequently utilized towards the development of a systematic, computationally effective method for computing the optimal scheduling policy for any given CRL configuration. While too complex for practical implementation, such a method is instrumental for the design of novel approximating schemes to the optimal scheduling policy, since it allows the experimental assessment of their quality.

3. The penultimate objective of the presented research program is to identify a systematic, efficient and scalable approximating scheme for the optimal scheduling policy characterized in Task 1 and 2. The quality of the obtained approximations will be experimentally assessed by "benchmarking" it against the optimal scheduling policy. This benchmarking activity will be performed in the context of small configurations where the optimal policy is effectively computable through the approach to be developed in Task 2.

The obtained results with respect to these objectives are reported in the rest of this document and they have also partially appeared in [10, 9]. A summary of these results is as follows: A formal system representation for the capacitated re-entrant line, based

on the framework of *Generalized Stochastic Petri nets* (*GSPNs*) [2] was developed and it was shown that it (i) allows the seamless integration of logical/structural and timed-based aspects of the system behavior, (ii) provides an analytical formulation for the underlying scheduling problem, and (iii) leads to pertinent qualitative insights regarding the structure of the optimal scheduling policy. For the case of small-sized systems, the proposed framework supports the thorough characterization of the structure of the optimal policies under various system parameterizations, allowing, thus, for a more systematic study and a more profound understanding of the (timed) dynamics taking place in these environments. Furthermore, for the purpose of computational effectiveness, the relevant scheduling problem was transformed to an *Average Reward Markov Decision Process (AR-MDP)* problem, which can be constructed algorithmically for any given fab configuration, and when solved for a number of small system configurations, it provides the benchmark towards the subsequent development of scalable approximating scheduling methods. In addition, this MDP-based modeling and analysis suggests a potential approximating scheduling method that is based on the approximation of the *optimal relative value function* with a parameterized function that is polynomially evaluated for any given state. Indeed, such an approximating function was developed, based upon results and insights coming from (i) the relevant MDP approximating theory and (ii) queueing theory. The representational capability of the considered approximating structure and the performance of the resulting scheduling policy were evaluated through a numerical experiment. An additional development of this research program has been an analytical characterization of the optimality of various deadlock resolution strategies for Markovian resource allocation systems under the objective of maximizing throughput; this result has appeared in [42].

The rest of this document is structured as follows: The next chapter discusses the current state of the art in performance modeling, analysis, and control of re-entrant lines, and provides the necessary background for describing the contents and contributions of the proposed research program. Task 1 in our research objectives is the content of Chapter 3, while Task 2 is addressed in Chapter 4. Task 3 is the topic of Chapter 5. Chapter 6 addresses the aforementioned developments regarding the optimality of various deadlock

resolution strategies in Markovian resource allocation systems. Finally, Chapter 7 concludes the discussion developed in the previous chapters and suggests possible future work.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

2.1 Resource Allocation Systems and Deadlock Avoidance Policies

The flexibly automated manufacturing system as an RAS and the RAS deadlock [39, 44] A flexibly automated production system can be pertinently abstracted to a *Resource Allocation System (RAS)*, consisting of a set $\mathcal{R} = \{R_i,\ i = 1, \ldots, m\}$ of *resource types*, and a set $\mathcal{J} = \{JT_j,\ j = 1, \ldots, n\}$ of *job / process types*, that can be executed in the system through sequential allocation of the system resources. Each resource type R_i is further characterized by its capacity C_i, i.e., a *finite* integral number indicating the number of units of this particular resource possessed by the system. Furthermore, resources are *reusable*, i.e., their allocation and deallocation to the system processes do not alter them in any way; in that sense, they constitute a system *invariant*. Jobs are executed in the system through a series of *(processing) stages*, and therefore, each job type JT_j is defined by a stage sequence: $JT_j =< JT_{jk},\ k = 1, \ldots, l(j) >$. In addition, each job stage JT_{jk} is further characterized by a resource allocation vector $A_{jk} \in (Z^+)^m$, indicating the number of resource units from each resource type that is required for the successful execution of the stage.

In the context of flexibly automated manufacturing systems, and the underlying RAS, *deadlock* arises due to the fact that a job, having finished the execution of a certain stage JT_{jk}, releases the resources allocated to it for the support of this stage, only after it has secured – i.e., been allocated – the resources for the execution of the successive stage $JT_{j,k+1}$.[1] This *"hold while waiting"* effect, combined with the exclusive and non-preemptive allocation of the finite system resources to the running jobs, can give rise to circular-waiting patterns,

[1] A typical example for this phenomenon is the allocation of the system buffering capacity available at the various workstations and material handling units. Being a physical entity, a job must always be accommodated somewhere.

in which a set of jobs is permanently blocked, since each of them, in order to proceed, requires the allocation of some resource unit(s) currently held by some other job in the set. In most manufacturing system contexts, the occurrence of a deadlock is a major disruption, since, the deadlocked jobs will not be able to advance and finish through "normal" system operation, and, while the deadlock persists, the effective utilization of the resources involved is equal to zero. Furthermore, the deadlock resolution will typically require external (human) intervention, and the transfer of unfinished jobs to temporary storage.

RAS logical/structural analysis and deadlock avoidance From an analytical / methodological standpoint, the deadlock problem is systematically addressed by modeling the behavior of the considered RAS as a *Finite State Automaton (FSA)* [7]. An *event*, $e \in E$, of this FSA, corresponds to the advancement of any job in the system by one stage / step. The RAS *state*, $s \in S$, is defined by the distribution of the currently running jobs to the various processing stages supported by the system. The automaton *state transition function*, $f : S \times E \rightarrow S$, is a formal expression of the aforementioned resource allocation mechanism: $f(s, e)$ is mapped to the resulting state s', if the job step defined by event e is *feasible* under the resource allocation described by state s; otherwise, it is mapped back to state s. The *initial* and *final* states of this automaton correspond to state s_0, denoting the state in which the system is idle and empty of any jobs. As a result, the *language* accepted by this automaton corresponds to complete production runs. Finally, we notice that this FSA model can be expressed graphically by its *State Transition Diagram (STD)*, i.e., a graph with *nodes* corresponding to the FSA states, and *arcs* corresponding to the feasible state transitions.

In order to characterize the deadlock problem arising in sequential RAS and the relevant notion of deadlock avoidance, we must first characterize the *feasible* behavior generated by the considered RAS, without any external supervision. Hence, the uncontrolled system behavior is characterized by the FSA *reachable* subspace, S_r, consisting of all states $s \in S$ for which there exists a feasible sequence of transitions from s_0 to s. Formally, this definition of S_r is denoted by $S_r = \{s \in S : s_0 \xrightarrow{*} s\}$. Then, the deadlock development in the operation

of the uncontrolled system leads to the formation of *strongly connected components* in S_r, from which, however, the empty state, s_0, is not reachable through any sequence of feasible transitions. By this token, a *correct Deadlock Avoidance Policy (DAP)*, $\mathcal{P}$, tries to restrict the system operation to a *strongly connected component of* S_r *which contains the empty state* s_0. Let us denote the subspace *admissible* by DAP $\mathcal{P}$ by $S_r(\mathcal{P})$. Given an RAS configuration, an applied DAP is characterized as *optimal*, if the corresponding admissible subspace is the *maximal* strongly connected component of S_r which contains the empty state s_0. The set of states admitted by the optimal DAP, $\mathcal{P}^*$, will be characterized as (the set of) *reachable safe* states, and it will be denoted by S_{rs}. The complement of S_{rs} with respect to S_r is denoted by S_{ru}, and it constitutes the system *reachable unsafe* region; formally, $S_{ru} = S_r \backslash S_{rs}$. In the context of the considered RAS's, the optimal DAP, $\mathcal{P}^*$, is well-defined and effectively computable, but its computation is an NP-Hard problem [43]. Nevertheless, to facilitate the subsequent discussion, we shall assume that the deadlock avoidance strategy implements the *optimal* DAP, $\mathcal{P}^*$. We notice, however, that the derived results will also apply in the case that the optimal DAP is substituted by some other suboptimal policy that seeks to constrain the system to a strongly connected component of the safe region that further contains the empty state s_0; Such suboptimal DAPs are reported in [45].

2.2 Generalized Stochastic Petri Net-based Performance Evaluation

Performance evaluation using Timed PN Since their inception, timed Petri Net (PN) models, in general, and Generalized Stochastic Petri Net (GSPN) models, in particular, have been strongly advocated for the modeling and performance evaluation of production systems, primarily due to the clarity and specificity of their semantics, that provides a fairly simple and flexible interface between the production systems design and control problem, and the underlying concepts and results borrowed from the more general theory of stochastic systems. We refer the reader to [55, 14] for an extensive coverage of the use of timed PN models for manufacturing system modeling and analysis until the middle 1990's. More recently, the works of [59, 21, 60] have used timed PN's for the performance evaluation of

the production activity taking place in the more specific context of modern semiconductor manufacturing facilities. The first of these works [59] discusses the fundamentals for the modeling and performance evaluation of the fab operations through timed PN's, in the context of a broader survey on the (potential) use of the PN modeling framework in semiconductor manufacturing. The work of [21] adjusts to a class of Markovian Timed PN's modeling the operations of a semiconductor manufacturing re-entrant line, a methodology for the computation of performance bounds, that was originally developed in [26] for multiclass queueing networks. Finally, the work of [60] uses ideas and results from the theory of deterministic marked graphs [47] in order to compute performance measures for various cluster tool [48] configurations, operated under pre-specified scheduling policies.

This section provides a brief introduction to the modeling framework of *Generalized Stochastic Petri nets* (*GSPN*) [1, 2], and the way in which it supports the performance evaluation of the modeled system. In the following discussion it is assumed that the reader is familiar with the basic Petri net structural concepts, and the emphasis is placed on the modeling elements and techniques concerning the time-related aspects of the system behavior. We refer the reader to [34] for an introduction to the basic PN theory.

The GSPN modeling framework According to [2], a Generalized Stochastic Petri net (GSPN) is defined as a PN $\mathcal{N}=(P, T, W, M_0)$ with its transition set T partitioned into two sub-sets T_I and T_T, defining respectively the set of *immediate* and *timed* transitions. Immediate transitions fire in zero time, once they are enabled, whereas, timed transitions fire after a random, *exponentially* distributed, enabling time. Hence, in order to complete the formal definition of a GSPN $\mathcal{N}$, transitions $j \in T_T$ are associated with a (possibly marking-dependent) *firing rate*, r_j, that constitutes the defining parameter of the corresponding exponential distribution.

The above characterization of immediate and timed transitions implies that in a net reachable marking, m, where, both immediate and timed transitions are enabled, immediate transitions have precedence over the timed ones (since they fire instantaneously). Furthermore, such a marking m has zero duration in the net dynamics, and therefore, it is

characterized as *vanishing*. On the other hand, a marking m in which all enabled transitions are timed transitions, has an expected duration $E(m) = 1/\sum_{j \in T_T(m)} r_j$, where $T_T(m)$ denotes the set of enabled timed transitions in marking m; therefore, such a marking is characterized as *tangible*. In the following, given a GSPN net $\mathcal{N}$ with initial marking M_0, the set of reachable tangible markings will be denoted by $R_T(\mathcal{N}, M_0)$ and the set of reachable vanishing markings will be denoted by $R_V(\mathcal{N}, M_0)$. Obviously, the set of reachable markings $R(\mathcal{N}, M_0) = R_T(\mathcal{N}, M_0) \cup R_V(\mathcal{N}, M_0)$.

The exponential nature of the firing times of the transitions enabled in a tangible marking m defines also an *arbitration* mechanism for their firing, i.e., each of these transitions will fire with probability $r_j / \sum_{j \in T_T(m)} r_j$. On the other hand, in a vanishing marking with more than one enabled immediate transitions, the contest of these transitions for firing must be arbitrated through some externally imposed logic. Specifically, given a marking m with a set of simultaneously enabled immediate transitions, $I(m)$, the modeler must provide a probability distribution regulating the firing of the transitions in $I(m)$. In the GSPN terminology, this probability distribution is characterized as a *random switch* $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{I(m)}\}$. Furthermore, if the random switches regulating the net behavior are marking-dependent, they are characterized as *dynamic*; otherwise, they are *static*.

According to [2], the motivation for the introduction of the immediate transitions in the net model is the desire to focus the time-related characterization of its behavior on those activities that have the significantly longest durations, and therefore, the strongest impact on the system performance. By placing the emphasis on the events with the longest timings and their associated tangible markings, the computational cost for the performance evaluation of the considered system is significantly reduced. We believe that another important modeling feature, resulting from the presence of immediate transitions in the GSPN modeling framework, is the ability to naturally separate the modeling of the control function from the modeling of the net dynamics corresponding to the various physical processes. Specifically, the various control commands imposed on the underlying (production) system are *logical events* which can be modeled by immediate transitions, whereas, the events corresponding to processing, transport or staging activities resulting from the physical execution

of these commands, are more appropriately modeled by timed transitions.

Performance evaluation of GSPN models The limitation of the timing models regulating the firing of the net transitions to immediate transitions and transitions with exponentially distributed enabling time, implies that the stochastic process modeling the time-based behavior of a GSPN $\mathcal{N}$ is a *semi-Markov process* [2] with a discrete state space, S, given by the net reachability space $R(\mathcal{N}, M_0)$. In particular, the *untimed* system dynamics, defined by its transitional patterns among the various states of its reachable state space, are characterized by the, so called, *Embedded Markov Chain* (*EMC*), whose transition probability matrix $Q = [q_{kl}]$ is determined by the externally specified random switches, in the case of vanishing markings, and the *exponential race* of the enabled events, in the case of tangible markings. If this EMC is finite-state, homogeneous and irreducible, it possesses a steady-state distribution $\mathbf{y}$, obtained by the system of equations

$$\mathbf{y} = \mathbf{y}Q \; ; \sum_{m_k \in R(\mathcal{N}, M_0)} y_k = 1. \tag{1}$$

The availability of the EMC steady-state distribution, $\mathbf{y}$, subsequently allows the computation of the steady-state probabilities, π_k, characterizing the timed system behavior, through the following formula [2]:

$$\pi_k = \begin{cases} 0, & m_k \in R_V(\mathcal{N}, M_0) \\ \frac{y_k E[m_k]}{\sum_{m_l \in R_T(\mathcal{N}, M_0)} y_l E[m_l]}, & m_k \in R_T(\mathcal{N}, M_0). \end{cases} \tag{2}$$

Notice that, as expected, the steady-state probability, π_k, is equal to zero for all reachable vanishing markings $m_k \in R_V(\mathcal{N}, M_0)$. On the other hand, the second branch of Equation 2 indicates that the percentage of time that the system spends in a reachable tangible marking m_k, is a function of the relative frequency with which this state is visited, determined from the untimed system dynamics, and the expected times that it spends in each reachable tangible marking. Once the steady-state probability vector π has been obtained, various performance measures of interest, characterizing the long-run system behavior, can be defined as appropriate functions of π and the other system parameters.

Extending the GSPN-based modeling to systems with non-Markovian dynamics We conclude this section with another remark regarding the modeling and analytical power of GSPN's. As it was mentioned above, the ability to evaluate analytically the steady-state probabilities characterizing the long-run system behavior is established on the requirement that all the distributions regulating the firing of the net timed transitions are of exponential type. This would seem quite restrictive since in many "real-life" environments, including those related to semiconductor manufacturing, the actual event timings might not be exponentially distributed. However, this restriction can be circumvented, whenever the exponential distribution is deemed as a too unrealistic assumption, by substituting each timed transition in the original GSPN model with a GSPN subnet modeling a *phase-type* distribution, that approximates the timing distribution of the replaced transition to any desired degree of accuracy, without compromising the GSPN structure of the final model. We refer the reader to [36] for a detailed treatment of phase-type distributions and the relevant approximation theory.

2.3 Markov Decision Processes and Approximating Scheduling Policies using Value Function Approximation

This section provides a brief introduction to *Markov Decision Processes* (MDP) that constitutes another formal framework for the performance analysis and control of multi-class queueing networks and resource allocation systems. As it is shown in the following, beyond providing a systematic characterization of optimality and the structure of optimal policies, this framework can also constitute the basis for the development of effective approximating techniques. The connection between the MDP and the previously discussed GSPN modeling framework, in the context of the considered application, is systematically investigated in Chapter 4.

Markov Decision Process-based performance control According to [38, 4], an MDP problem is defined by a 4-tuple $(S, U, \widetilde{P}, r)$, where S is a set of states, U is a set of state-dependent actions, $\widetilde{P}$ is a state transition probability matrix, and r is a cost/reward function associating each state-control pair with a cost/reward. More specifically, the system is

modeled via a set of states S, which evolve stochastically over time, under the influence of a control action sequence in U, that is selected based on the current state information at each decision epoch, and intends to optimize some performance objective(s). Since the emphasis of this work is on the maximization of the (long-term) system throughput, next we focus on a particular class of MDP problem, known as the *average reward* Markov Decision Process problem (with finite state and control space). This problem can be formalized as follows:

We consider a discrete-time dynamic system evolving according to

$$i_{t+1} = f(i_t, u_t, w_t), \tag{3}$$

where i_t is a system state, u_t is a control decision, and w_t is some experienced disturbance/random element at time t. A function $r : S \times U \rightarrow R$ associates a reward $r(i_t, u_t)$ with a decision u_t made at state i_t. A *stationary policy* μ is a mapping $\mu : S \rightarrow U$ with $\mu(i)$ being the control action to be taken at state i. For each such policy μ, the *average reward per stage*, which evaluates the expected future reward per stage as a function $J^\mu : S \rightarrow \mathcal{R}$ of the current state i, is defined by

$$J^\mu(i) = \lim_{N\rightarrow\infty} \frac{1}{N} E\Big[\sum_{t=0}^{N-1} r(i_t, \mu(i_t))|i_0 = i\Big]. \tag{4}$$

The *optimal* average reward per stage, J^*, is defined by

$$J^*(i) = \max_\mu J^\mu(i), \tag{5}$$

and an *optimal policy* μ^* is one that maximizes J^μ for any given state i, i.e.,

$$\mu^*(i) = \arg\max_\mu J^\mu(i). \tag{6}$$

A well known result in the MDP theory is that, under the assumption that every optimal stationary policy is *unichain* [38, 4], the optimal average reward per stage is independent of the initial state, and the corresponding optimal average reward per stage, λ^*, satisfies the following Bellman's equation:

$$\lambda^* + \tilde{h}^*(i) = \max_{u\in U(i)} \Big[r(i,u) + \sum_{j=1}^{n} \tilde{p}_{ij}(u)\tilde{h}^*(j)\Big], i = 1, \ldots, n. \tag{7}$$

In Equation 7, $\tilde{h}^*(i)$ is called the *relative value function* of state i, and it is the difference between the expected reward to reach a target state k from state i for the first time and the reward that would be incurred if the reward per stage was the average λ^* [38, 4].[2] U(i) is a set of actions defined at state i. Furthermore, the optimal policy $\mu^*(i)$ at state i is defined as follows:

$$\mu^*(i) = \arg \max_{u \in U(i)} \left[r(i,u) + \sum_{j=1}^{n} \tilde{p}_{ij}(u) \tilde{h}^*(j) \right], i = 1, \ldots, n. \quad (8)$$

There have been many research works to address the MDP problem defined by Equations 7 and 8. In particular, Dynamic Programming has provided a variety of solution methods, including the *value iteration* method, the *policy iteration* method, and the *Linear Programming* method [38, 4]. All these methods try to compute the optimal relative value function and derive the corresponding optimal control policy through Equation 8; i.e., theoretically, an optimal policy can be found by first solving Bellman's equation 7, and then computing the *"greedy"* policy defined by the resulting optimal relative value function. We notice, however, that for most practical systems, the state space expands exponentially fast, which makes impossible the computation (or even storage) of the relative value function at each state, and necessitates the development of efficient and scalable approximating methodologies.

Approximating the optimal policy using relative value function approximation for the average reward MDP problem As mentioned above, even though Dynamic Programming can provide an analytical framework for addressing the re-entrant line performance control problem, there are still critical remaining issues arising from its very high computational complexity (*the curse of dimensionality*), which render the solution methods mentioned above inapplicable to most realistic fab scheduling problems. As a result, usually simplified analyses or *heuristics* are applied.

During the past few years, there has been an alternative emerging theory to address these computational issues, which is called *Neuro-Dynamic Programming* (NDP) [5]. In the

[2] We notice, for completeness, that Bellman's equation defines the optimal relative value function upto *translation* only; i.e., if $\tilde{h}^*(i)$ is an optimal relative value function, then every other function $\tilde{h}'(i) = \tilde{h}^*(i) + c$ satisfies Equation 7, $\forall\ c \in R$.

NDP framework, a *feature*-based compact representation is used in order to generate an effective approximation of the optimal value function, and the corresponding greedy policy is adopted as a near-optimal control policy. In the literature, there are two prevailing approximation schemes for computing approximated value functions, including: (i) *look-up table* methods, and (ii) *parametric representation* methods. Look-up table methods [51, 5] essentially aggregate the underlying state space to a number of aggregate components on the basis of a preselected set of features, and they maintain one value for each aggregate component. Parametric representation methods [51, 53, 5, 13] use a functional approximation of the optimal value function, constructed as a linear combination of a set of *"basis"* or *"feature"* functions, and select the *parameters/weights* of the approximation by using some Dynamic Programming (DP)-based methods, simulation, or *reinforcement learning* algorithms such as Temporal-Difference (TD) learning [53, 52]. We notice that, in these methods, the number of parameters grows only linearly with the number of the employed feature functions, and therefore, it is important to select a rich and pertinent set of feature functions that will (have the potential to) result in high quality approximation. A comprehensive survey of these methods applied to both *discounted* and *average reward* MDP problems, can be found in [5].[3]

This research program focuses on the parametric representation methods for synthesizing approximating policies for the considered average reward MDP problem. The potential representational capability of those approaches is documented in several notable success stories in the literature, which include a program that plays Backgammon [50] and the case studies reported in [58, 54]. It is also known, though, that two critical issues for the success of these methods are (i) the selected set of feature functions and (ii) the applied tuning algorithms. Selecting a good set of feature functions is, to a large extent, application driven, and therefore, it needs to be investigated on a case by case basis, while taking into consideration the idiosyncrasies of the problem addressed. On the other hand, for any given policy, μ, the weight-tuning algorithms seek to iteratively update the employed

[3]We notice, however, that in case of the average reward MDP problems, those methodologies and the underlying theory are not as complete as for the discounted problems.

weight vector, based on information drawn from either simulation or observation of the corresponding Markov Process, with the objective of developing a good approximation of the relative value function of μ, as time progresses. Presumably, these tuning algorithms can be combined with other DP-based algorithms such as the *(approximate) value iteration* method, or the *(approximate) policy iteration* method, for the eventual derivation of an approximating control policy. However, the overall convergence of such a computational scheme, for the average reward MDP problem, has been a challenging issue, and while some results are available [52], the design of effective tuning algorithms for this class of problems is open to further investigation.

2.4 Re-entrant Line Scheduling Problems

Currently, there are many published works dealing with the scheduling problems arising in semiconductor manufacturing systems [57, 56, 23, 24, 31, 27, 28], where the system abstraction employed is the *re-entrant line* discussed in Chapter 1. These works have provided a variety of scheduling policies for the underlying scheduling problems, most of which are based on heuristics because of the complexity of the process flow materialized by the re-entrant line and the stochastic variability involved. According to [28], these scheduling policies can be classified into (i) work release policies and (ii) sequencing policies, and collectively, they seek to optimize some performance measure(s), including (i) throughput rate, (ii) cycle time, (iii) work-in-process (WIP) inventories , and (iv) throughput capacity, i.e., the maximum sustainable throughput rate of a fab operating under a given scheduling policy. This section discusses briefly some representative scheduling policies currently considered for re-entrant line scheduling problems.

First of all, work release policies [56, 28], attempting to regulate the work flow into the system in order to optimize the throughput or cycle time, include the *deterministic release*, the *CONWIP*, and the *Closed Loop release* policies. In [56], the performance of these workload policies is tested for some semiconductor fab models, while combined with other sequencing policies, which decide which of the jobs waiting for processing at each station, is to be processed next. Such sequencing policies include the *First In First Out (FIFO)*

policy, the *Shortest Processing Time* policy, the *Shortest Remaining Time* policy, the set of *Least Slack* policies [31, 23, 22, 27, 57], that try to reduce the mean and variance of job cycle time by selecting the most urgent jobs, and *buffer priority* policies [30, 22, 25]. Most of the aforementioned policies were developed in a heuristical manner, and they attempt to control either (i) the WIP inventory and/or (ii) the material flow within the semiconductor fab, in order to optimize some of the performance measures mentioned above. Some more recent significant research works deal with the scheduling problems in re-entrant lines based on the multi-class queueing network theory, the fluid model theory, or the MDP theory. In [33, 8], the convergence of policy iteration and value iteration algorithms for the average cost problem is addressed, and these algorithms are applied for the scheduling problem of multi-class queueing networks modeling re-entrant lines. [20] introduces a 2-parameter (queueing) network and an efficient scheduling policy for it is synthesized by considering a more tractable workload model, using a "fluid" model-based approximation. [12] proposes a family of "maximum pressure" policies and proves that a network operating under such a policy achieves a maximum throughput predicted by an LP characterizing the system bottleneck(s); the results are applied to re-entrant lines. Similarly, [32] describes some approaches to the synthesis of optimal policies for multi-class queuing networks based on queueing networks and fluid models, and applies them on the re-entrant line scheduling problem.

As mentioned in Chapter 1, the re-entrant line model employed in these research works considers only the job contest for the finite processing capacity of the system workstations, ignoring completely the effects and complications arising from additional operational issues like the finite buffering capacity of the system workstations/production units. This element invalidates the direct translation of these past results to the capacitated re-entrant line context considered in this work. However, the findings and the insights of these works can potentially provide useful guidelines in the selection of the features to be employed in the approximation of the relative value function discussed in Section 2.3.

CHAPTER 3

GENERALIZED STOCHASTIC PETRI NET-BASED PERFORMANCE ANALYSIS AND CONTROL OF CAPACITATED RE-ENTRANT LINES

3.1 The Capacitated Re-entrant Line and its GSPN-based Modeling

The capacitated re-entrant line The capacitated re-entrant line considered in this research program refines the basic re-entrant line model, presented in Chapter 1, through the explicit modeling of (i) the workstation buffering capacity and its internal material flow, and (ii) the interconnecting material handling system. More specifically, it is assumed that each workstation W_i, $i = 1, \ldots, L$, consists of B_i buffer slots and S_i identical servers. 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. Once in the station local buffer, the part competes for one of the station servers for the execution of the requested stage. Under the current model definition, it can be assumed either that the part is mounted into the server for its processing and then it is returned to its designated slot, or that the server processes the part by visiting the corresponding buffer. Based on this description of the workstation operation, it is natural to assume that $S_i \leq B_i, \forall\ i$. A part having finished the processing of its current stage at a certain station, waits in its allocated buffer for transfer to the next requested station. This transfer is facilitated by the central (automated) material handling system, however, due to the finite buffering capacity, it should be authorized by a *structural control policy* (SCP) [45], 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 subsequent analysis, the central material handling system can

be considered to be either a centrally located robotic manipulator, or a single-loop AGV system; in the former case, the re-entrant line is the modeling abstraction for what is known as a cluster tool, while in the latter case, the resulting model represents the dynamics of a modern fab bay, where the various process tools possess a local stocker of limited buffering capacity.

Following the typical practice, the main scheduling objective considered in the undertaken analysis is the maximization of the long-run system throughput, and therefore, it is assumed that there exists an infinite amount of raw material waiting for processing at the line's Input/Output (I/O) station. Furthermore, in order to facilitate the subsequent modeling and analysis, it is also assumed that all stage processing and transfer times are exponentially distributed. In particular, the processing time of stage J_j, $j = 1 \ldots, M$, is assumed to follow an exponential distribution with finite non-zero rate μ_j, while job transfer times are assumed to be exponentially distributed with non zero rate α, that applies uniformly across all the transferring operations. This presumed uniformity of the mean transfer times is introduced in order to simplify the computations involved in the presented example, and it also allows the analytical investigation of the limiting case where the transfer times are negligible with respect to the processing times involved, by taking $\alpha \to \infty$ in the derived expressions. Finally, we notice that the rather unrealistic assumption of exponentially distributed processing and transfer times can be eventually relaxed by applying a phase-type distribution that approximates the original/empirical distribution of the corresponding event timing as discussed before.

Example The above general description of the capacitated re-entrant line is exemplified by the small system presented in Figure 1. The depicted configuration possesses two stations, W_1 and W_2, with $S_1 = S_2 = 1$ and $B_1 = 1$; $B_2 = 2$. Furthermore, the supported production sequence is $J =< J_1, J_2, J_3 >$, with $W(J_1) = W(J_3) = W_1$ and $W(J_2) = W_2$. Finally, stage processing times are exponentially distributed with rates $\mu_j > 0$, $j = 1, 2, 3$,

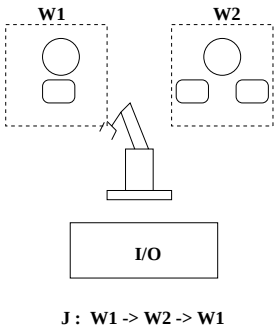


Figure 1: Example: The capacitated re-entrant line

and so are the involved transfer times, with a uniform rate α. Then, the underlying scheduling problem is to determine how to allocate the workstation processing and buffering capacity to the job stages competing for it, in order to maximize its long-run / steady-state throughput, while maintaining logical correctness of material flow, i.e., deadlock-free operations. For this small configuration, it is easy to see that, under the operational assumptions outlined above, the system material flow will remain deadlock-free, as long as

$$|J_1| + |J_2| \leq B_1 + B_2 - 1 = 2, \tag{9}$$

where $|J_j|$, $j = 1, 2, 3$ denotes the number of job instances in $W(J_j)$ executing stage J_j. $\diamondsuit$

GSPN-based modeling of capacitated re-entrant lines The GSPN modeling the behavior of the capacitated re-entrant line of Figure 1, under the control of the maximally permissive SCP of Equation 9, is depicted in Figure 2. Specifically, in the GSPN of Figure 2, the part flow dynamics associated with each processing stage J_j, $j = 1, 2, 3$, are modeled by the corresponding net path $< T_{ja}, P_{jt}, T_{jt}, P_{ji}, T_{jl}, P_{jp}, T_{jp}, P_{jo}, T_{jd} >$, while it also holds $T_{jd} \equiv T_{j+1,a}$, with $j = 4$ denoting the last unloading step. A token in place P_{jt} represents a part in transit to the buffer of workstation $W(J_j)$; a token in place P_{ji} represents a part in the buffer of $W(J_j)$ waiting the allocation of one of the buffer servers; a token in place P_{jp} represents a part in processing of stage J_j; finally, a token in place P_{jo} represents a part having finished processing of stage J_j, and waiting for transfer to the next requested workstation or, in case that J_j is the last processing stage, to the I/O station.

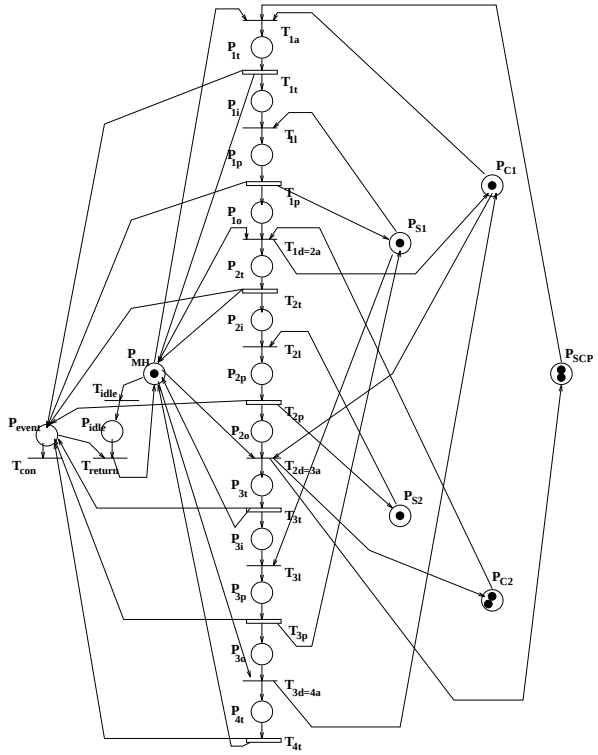


Figure 2: Example: The GSPN model

On the other hand, places P_{MH}, P_{S_i}, P_{B_i}, $i = 1, 2$, and P_{SCP} model respectively the availability of the system transporter, workstation servers and buffers, and the logic of the applied SCP, according to the standard, by now, modeling practice of resource-process nets [3]. It is important to notice that transitions T_{ja}, T_{jl} and T_{jd}, that are associated with the various decisions regarding the allocation of the system buffering, processing and/or transport capacity, are untimed / immediate transitions, while the delays experienced from the processing and/or transfer times involved with the execution of these decisions, are modeled by the timed transitions T_{jt} and T_{jp}. This separation of the net components modeling the timings of the various system events from the net structure modeling the underlying resource allocation and the associated decision making, enables the modeling of timing distributions other than exponential through the (local) substitution of the corresponding timed transitions by GSPN subnets modeling the approximating phase-type distributions. It also allows, as it is shown below, the modeling of the required scheduling logic through

a set of *dynamic random switches*, that resolve the conflicts among the immediate transitions that are simultaneously enabled at the net reachable vanishing markings. Finally, some explanation is necessary about the role of places P_{idle}, P_{event} and their associated transitions T_{idle}, T_{return} and T_{con}. This subnet essentially establishes a GSPN-compatible mechanism for representing some deliberate idleness of the system transporter in the underlying scheduling logic, since, in the considered operational context, the optimal scheduling policy is not necessarily non-idling.[1] Hence, the triggering of transition T_{idle} consumes the transporter-modeling token, which remains in place P_{idle}, until the immediate transition T_{return} is enabled through the presence of a token in place P_{event}. P_{event} is marked every time that one of the system timed transitions fires, signaling the completion of some event. Notice that T_{return} will always be in conflict with transition T_{con}, but it is assumed to have priority over the latter, which is technically imposed by setting the corresponding (static) random switch to $\{\xi_{T_{return}} = 1,\ \xi_{T_{con}} = 0\}$. Finally, T_{con} is a sink transition that "consumes" event completion signaling tokens, in case that the transporter is not (deliberately) idling.

GSPN-based performance evaluation of capacitated re-entrant lines In the case of GSPN's modeling the behavior of capacitated re-entrant lines, the underlying EMC is finite-state and homogeneous, but it might contain absorbing states due to the presence of transition T_{idle}. Specifically, if T_{idle} fires while no other event is in process, the token representing the system transporter will be permanently stuck in place P_{idle}. This problem can be addressed by disabling these problematic firings of T_{idle} through appropriate setting of the corresponding dynamic random switches. The resulting modified EMC has the property that from every pair of states s_i and s_j, there exists a deterministic scheduling policy that renders s_j accessible from s_i.[2] This property subsequently guarantees the existence of an optimal pricing of the random switching probabilities, ξ_l, appearing in the modified EMC,

[1] In fact, a similar remark applies for the idleness of the workstation servers; however, in this example, the deliberate idleness of workstation servers is not considered explicitly since it will always be sub-optimal for the given system configuration.

[2] This is the effect - in fact, the objective - of the applied structural control policy.

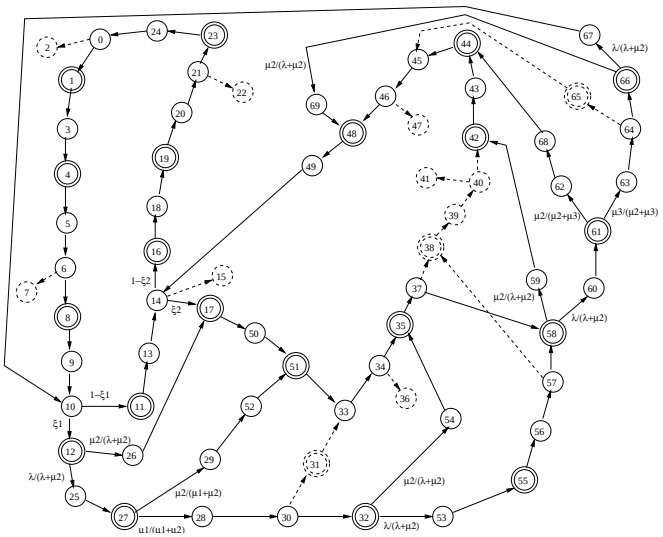


Figure 3: Example: The Embedded Markov Chain (EMC)

that leads to a controlled system behavior that is modeled by a *unichain* Markov chain, i.e., a Markov chain consisting of a single communicating class and possibly a set of transient states (c.f. [38], Section 8.3); in the following, the scheduling policies resulting from such pricings will be referred to as *unichain* policies, and their set will be denoted by UP. Constraining the search for an optimal scheduling policy in set UP, and letting $\overline{Q}(\xi)$ denote the transition probability matrix (TPM) of the aforementioned modified EMC, resulting from the removal of all the absorbing states, we obtain the following MP formulation for the problem of throughput maximization for a capacitated re-entrant line:

$$\max_{\xi \in \mathbf{UP}} \; TH(\xi) \equiv \sum_{(k,j)} \pi_k r_j I \quad \begin{array}{l} \{\text{Transition } T_j \text{ enabled in } m_k \\ \qquad \wedge \text{ cor. to an unloading event}\} \end{array} \tag{10}$$

s.t.

$$\mathbf{y} = \mathbf{y}\overline{Q}(\xi) \; ; \; \sum_{m_k \in R(\mathcal{N}, M_0)} y_k = 1 \tag{11}$$

$$\forall \; k, \pi_k = \begin{cases} 0, & m_k \in R_V(\mathcal{N}, M_0) \\ \dfrac{y_k E[m_k]}{\sum_{m_l \in R_T(\mathcal{N}, M_0)} y_l E[m_l]}, & m_k \in R_T(\mathcal{N}, M_0) \end{cases} \tag{12}$$

$$\forall \; l, \; \xi_l \geq 0 \tag{13}$$

$$\forall \text{ random switch } \Xi_u, \sum_{l:\xi_l \in \Xi_u} \xi_l = 1. \quad (14)$$

Equations 11 and 12 characterize the timed system behavior by computing the steady-state probabilities as discussed in Section 2.2. Equations 13 and 14 define the probability distributions of the random switches associated with the various vanishing markings/states.

Example The EMC for the GSPN of Figure 2 is presented in Figure 3, while the net markings corresponding to the various states depicted in Figure 3 are listed in Table 1. In Figure 3, states corresponding to vanishing markings are depicted by single circles, while states corresponding to tangible markings are depicted by double circles. Furthermore, the part of the chain depicted in dashed lines should be inaccessible under operation by any optimal scheduling policy, either because it leads to dead/absorbing states (c.f. the relevant discussion above), or because the transitions branching to that part of the chain essentially introduce some unnecessary delay in the system operation, by deliberately idling the server. As a more concrete example of the latter case, consider state s_{30} in Figure 3, which, according to Table 1, corresponds to a state where a job, j_1, in workstation W_1, having finished processing of stage J_1 requests transfer to workstation W_2, that currently contains only another job, j_2, in processing of its second stage. Moreover, the system transporter is available, and it is easy to check that the requested transfer is physically feasible and admissible by the applied SCP. Under these circumstances, deliberately idling the transporter, by firing transition T_{idle}, will definitely be a suboptimal decision, since the only way that the system can progress once job j_2 has completed the execution of its current stage, is by eventually executing the postponed transfer of job j_1 to W_2, and the overall operation of the system will have been slowed down by the corresponding unnecessary delay. The remaining modified EMC, depicted with solid lines in Figure 3, contains only two random switches of two options each, which combined with Equation 14, leaves us with two decision variables ξ_1 and ξ_2. Finally, the reader can verify that any pricing $(\xi_1, \xi_2) \in [0, 1]^2$ leads to unichain behavior for the controlled system. $\diamondsuit$

Table 1: Example: The EMC markings

s_k	$P_{1t}P_{1i}P_{1p}P_{1o}$	$P_{2t}P_{2i}P_{2p}P_{2o}$	$P_{3t}P_{3i}P_{3p}P_{3o}P_{4t}$	$P_{MH}P_{idle}P_{event}$	$P_{S_1}P_{S_2}$	$P_{B_1}P_{B_2}P_{SCP}$
0	0 0 0 0	0 0 0 0	0 0 0 0 0	1 0 0	1 1	1 2 2
1	1 0 0 0	0 0 0 0	0 0 0 0 0	0 0 0	1 1	0 2 1
2	0 0 0 0	0 0 0 0	0 0 0 0 0	0 1 0	1 1	1 2 2
3	0 1 0 0	0 0 0 0	0 0 0 0 0	1 0 1	1 1	0 2 1
4	0 0 1 0	0 0 0 0	0 0 0 0 0	0 1 0	0 1	0 2 1
5	0 0 0 1	0 0 0 0	0 0 0 0 0	0 1 1	1 1	0 2 1
6	0 0 0 1	0 0 0 0	0 0 0 0 0	1 0 0	1 1	0 2 1
7	0 0 0 1	0 0 0 0	0 0 0 0 0	0 1 0	1 1	0 2 1
8	0 0 0 0	1 0 0 0	0 0 0 0 0	0 0 0	1 1	1 1 1
9	0 0 0 0	0 1 0 0	0 0 0 0 0	1 0 1	1 1	1 1 1
10	0 0 0 0	0 0 1 0	0 0 0 0 0	1 0 0	1 0	1 1 1
11	0 0 0 0	0 0 1 0	0 0 0 0 0	0 1 0	1 0	1 1 1
12	1 0 0 0	0 0 1 0	0 0 0 0 0	0 0 0	1 0	0 1 0
13	0 0 0 0	0 0 0 1	0 0 0 0 0	0 1 1	1 1	1 1 1
14	0 0 0 0	0 0 0 1	0 0 0 0 0	1 0 0	1 1	1 1 1
15	0 0 0 0	0 0 0 1	0 0 0 0 0	0 1 0	1 1	1 1 1
16	0 0 0 0	0 0 0 0	1 0 0 0 0	0 0 0	1 1	0 2 2
17	1 0 0 0	0 0 0 1	0 0 0 0 0	0 0 0	1 1	0 1 0
18	0 0 0 0	0 0 0 0	0 1 0 0 0	1 0 1	1 1	0 2 2
19	0 0 0 0	0 0 0 0	0 0 1 0 0	0 1 0	0 1	0 2 2
20	0 0 0 0	0 0 0 0	0 0 0 1 0	0 1 1	1 1	0 2 2
21	0 0 0 0	0 0 0 0	0 0 0 1 0	1 0 0	1 1	0 2 2
22	0 0 0 0	0 0 0 0	0 0 0 1 0	0 1 0	1 1	0 2 2
23	0 0 0 0	0 0 0 0	0 0 0 0 1	0 0 0	1 1	1 2 2
24	0 0 0 0	0 0 0 0	0 0 0 0 0	1 0 1	1 1	1 2 2
25	0 1 0 0	0 0 1 0	0 0 0 0 0	1 0 1	1 0	0 1 0
26	1 0 0 0	0 0 0 1	0 0 0 0 0	0 0 1	1 1	0 1 0
27	0 0 1 0	0 0 1 0	0 0 0 0 0	0 1 0	0 0	0 1 0
28	0 0 0 1	0 0 1 0	0 0 0 0 0	0 1 1	1 0	0 1 0
29	0 0 1 0	0 0 0 1	0 0 0 0 0	0 1 1	0 1	0 1 0
30	0 0 0 1	0 0 1 0	0 0 0 0 0	1 0 0	1 0	0 1 0
31	0 0 0 1	0 0 1 0	0 0 0 0 0	0 1 0	1 0	0 1 0
32	0 0 0 0	1 0 1 0	0 0 0 0 0	0 0 0	1 0	1 0 0
33	0 0 0 1	0 0 0 1	0 0 0 0 0	0 1 1	1 1	0 1 0
34	0 0 0 1	0 0 0 1	0 0 0 0 0	1 0 0	1 1	0 1 0
35	0 0 0 0	1 0 0 1	0 0 0 0 0	0 0 0	1 1	1 0 0
36	0 0 0 1	0 0 0 1	0 0 0 0 0	0 1 0	1 1	0 1 0
37	0 0 0 0	0 1 0 1	0 0 0 0 0	1 0 1	1 1	1 0 0
38	0 0 0 0	0 0 1 1	0 0 0 0 0	0 1 0	1 0	1 0 0
39	0 0 0 0	0 0 0 2	0 0 0 0 0	0 1 1	1 1	1 0 0
40	0 0 0 0	0 0 0 2	0 0 0 0 0	1 0 0	1 1	1 0 0
41	0 0 0 0	0 0 0 2	0 0 0 0 0	0 1 0	1 1	1 0 0
42	0 0 0 0	0 0 0 1	1 0 0 0 0	0 0 0	1 1	0 1 1
43	0 0 0 0	0 0 0 1	0 1 0 0 0	1 0 1	1 1	0 1 1
44	0 0 0 0	0 0 0 1	0 0 1 0 0	0 1 0	0 1	0 1 1
45	0 0 0 0	0 0 0 1	0 0 0 1 0	0 1 1	1 1	0 1 1
46	0 0 0 0	0 0 0 1	0 0 0 1 0	1 0 0	1 1	0 1 1
47	0 0 0 0	0 0 0 1	0 0 0 1 0	0 1 0	1 1	0 1 1
48	0 0 0 0	0 0 0 1	0 0 0 0 1	0 0 0	1 1	1 1 1
49	0 0 0 0	0 0 0 1	0 0 0 0 0	1 0 1	1 1	1 1 1
50	0 1 0 0	0 0 0 1	0 0 0 0 0	1 0 1	1 1	0 1 0
51	0 0 1 0	0 0 0 1	0 0 0 0 0	0 1 0	0 1	0 1 0
52	0 0 1 0	0 0 0 1	0 0 0 0 0	1 0 0	0 1	0 1 0
53	0 0 0 0	0 1 1 0	0 0 0 0 0	1 0 1	1 0	1 0 0
54	0 0 0 0	1 0 0 1	0 0 0 0 0	0 0 1	1 1	1 0 0
55	0 0 0 0	0 1 1 0	0 0 0 0 0	0 1 0	1 0	1 0 0
56	0 0 0 0	0 1 0 1	0 0 0 0 0	0 1 1	1 1	1 0 0
57	0 0 0 0	0 0 1 1	0 0 0 0 0	1 0 0	1 0	1 0 0
58	0 0 0 0	0 0 1 0	1 0 0 0 0	0 0 0	1 0	0 1 1
59	0 0 0 0	0 0 0 1	1 0 0 0 0	0 0 1	1 1	0 1 1
60	0 0 0 0	0 0 1 0	0 1 0 0 0	1 0 1	1 0	0 1 1
61	0 0 0 0	0 0 1 0	0 0 1 0 0	0 1 0	0 0	0 1 1
62	0 0 0 0	0 0 0 1	0 0 1 0 0	0 1 1	0 1	0 1 1
63	0 0 0 0	0 0 1 0	0 0 0 1 0	0 1 1	1 0	0 1 1
64	0 0 0 0	0 0 1 0	0 0 0 1 0	1 0 0	1 0	0 1 1
65	0 0 0 0	0 0 1 0	0 0 0 1 0	0 1 0	1 0	0 1 1
66	0 0 0 0	0 0 1 0	0 0 0 0 1	0 0 0	1 0	1 1 1
67	0 0 0 0	0 0 1 0	0 0 0 0 0	1 0 1	1 0	1 1 1
68	0 0 0 0	0 0 0 1	0 0 1 0 0	1 0 0	0 1	0 1 1
69	0 0 0 0	0 0 0 1	0 0 0 0 1	0 0 1	1 1	1 1 1

3.2 Obtaining an Optimal Scheduling Policy

The solution of the MP formulation defined by Equations 10, 11, 12, 13 and 14 is a challenging problem because of (i) the non-linearity arising in Equations 11 and 12, and (ii) the additional requirement that $\xi \in UP$, which is necessary for the existence of the steady-state distribution implied by Equations 11 and 12. However, in this section, we establish that the considered formulation will always have an optimal solution which prices all primary decision variables, ξ_l, at one of their extreme values, 0 or 1, and therefore, it can be solved through enumerative techniques. From a modeling standpoint, such an optimal solution defines a *deterministic* scheduling policy. We notice that this finding is consistent with a more general result on the optimality of deterministic scheduling policies provided by the theory of MDP [38]; our work provides a specialization and a complete alternative derivation for it in the GSPN modeling framework. We proceed to this development through a series of lemmata.

Lemma 1 *The optimization problem defined by Equations 10, 11, 12, 13 and 14 can be transformed to an equivalent optimization problem of the form:*

$$\max_{\xi \in UP} \; TH(\xi) = \frac{N(\xi)}{D(\xi)} \tag{15}$$

s.t.

Equations (13) and (14),

where functions $N(\xi)$ and $D(\xi)$ are multi-linear[3] *in ξ. Furthermore, $D(\xi) \neq 0, \; \forall \; \xi \in UP$ satisfying Equations 13 and 14.*

Proof: Notice that, according to Equation 11, the variable vector $\mathbf{y}$, denoting the steady state probabilities of the net modified EMC, satisfies the linear system of equations:

$$\begin{bmatrix} \overline{Q}^T(\xi) \\ \mathbf{1}^T \end{bmatrix} \mathbf{y} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}, \tag{16}$$

[3] *i.e., first-degree polynomials with respect to each single variable ξ_l*

where **1** and **0** denote column vectors with all their elements equal to 1 and 0, respectively. Furthermore, the dynamic nature of random switches, assumed in this work, implies that each variable ξ_l appears in matrix $\overline{Q}^T(\xi)$ only once, namely in the column corresponding to the associated vanishing marking m. To facilitate the subsequent discussion, let us rewrite Equation 16 as

$$H\mathbf{y} = \mathbf{b}. \tag{17}$$

The ergodic nature of the modified EMC defined by the considered values of the variable vector ξ, implies that the linear system of Equation 17 has a unique solution, obtained by Crammer's rule [49]:

$$\forall\ m_k \in R(\mathcal{N}, M_0), \quad y_k(\xi) = \frac{\det(H_k(\xi))}{\det(H(\xi))}, \tag{18}$$

where matrix $H_k(\xi)$ is obtained from matrix $H(\xi)$ by replacing its k-th column by vector **b**. Furthermore, the fact that each variable ξ_l appears in a single element of matrix $H(\xi)$ implies that $\forall\ k,\ \det(H_k(\xi))$ is a multi-linear function in ξ. But then, Equation 12 implies that for all $m_k \in R_T(\mathcal{N}, M_0)$,

$$\pi_k = \frac{E[m_k]\det(H_k(\xi))}{\sum_{m_l \in R_T(\mathcal{N}, M_0)} E[m_l]\det(H_l(\xi))} = \frac{N_k(\xi)}{D(\xi)} \tag{19}$$

and $N_k(\xi)$ and $D(\xi)$ are multi-linear functions in ξ. The main result of Lemma 1 is obtained from Equation 19, by noticing that, according to Equation 10, $TH(\xi)$ is defined as the weighted sum of an appropriately selected set of π_k. The fact that $D(\xi) \neq 0$ over the considered feasible region, is established by the requirement that $\xi \in UP$, since it implies the existence of a limiting distribution for the continuous-time stochastic process modeling the time-based behavior of the controlled system. $\diamondsuit$

The next lemma establishes some additional structure for the polynomial functions $N(\xi)$ and $D(\xi)$, which is invoked in the proof of the theorem stating the main result of this section.

Lemma 2 *In the multi-linear functions $N(\xi)$ and $D(\xi)$ defined in Lemma 1, there are no products of variables ξ_l belonging in the same random switch Ξ_u.*

Proof: Remember that, according to the proof of Lemma 1, all variables ξ_l belonging to a single random switch Ξ_u regulating the transitions out of a vanishing marking m,

appear in the same column of matrix $H(\xi)$. Then, the truth of Lemma 2 follows from the elementary definition of the det() operator [49], and the definitions of functions $N(\xi)$ and $D(\xi)$ in the proof of Lemma 1. $\diamondsuit$

Theorem 1 *The MP formulation of Equations 15, 13 and 14, introduced in Lemma 1, will always have an optimal solution in which the primary decision variables, ξ_l, are priced in the set $\{0, 1\}$.*

Proof: Without loss of generality, suppose that each random switch Ξ_u has $|\Xi_u| \geq 2$. Then, solving the corresponding constraint in Equation 14 for one of the involved decision variables, to be denoted by $\xi_{i(u)}$, and replacing $\xi_{i(u)}$ in the objective function by the resulting expression, we can rewrite the formulation of Equations 15, 13 and 14 in a reduced variable space, as follows:

$$\max_{\xi \in UP} \; TH(\xi) = \frac{\hat{N}(\xi)}{\hat{D}(\xi)} \tag{20}$$

s.t.

$$\forall \text{ random switch } \Xi_u, \; \forall \; l \neq i(u), \; \xi_l \geq 0 \tag{21}$$

$$\forall \text{ random switch } \Xi_u, \sum_{l \neq i(u) : \xi_l \in \Xi_u} \xi_l \leq 1. \tag{22}$$

Lemma 2 implies that the functions $\hat{N}(\xi)$ and $\hat{D}(\xi)$ remain multi-linear polynomials in ξ. Then, the partial differentiation of function $TH(\xi)$ with respect to each variable ξ_l reveals that the objective function defined by Equation 20 is monotone with respect to every single variable ξ_l.

This monotonicity property of $TH(\xi)$, combined with the fact that for any $\xi \in UP$ such that $\forall \; l, \; \xi_l \in (0, 1)$, $\exists \; \delta > 0$ such that $\forall$ unit radius $\mathbf{r}$, $\xi + \delta \mathbf{r} \in UP$, further implies that there exists an optimal solution of the formulation defined by Equations 20, 21 and 22 that lies on the boundary of its feasible region. Hence, any such optimal solution $\xi^* \in UP$ must bind at least one of the Constraints 21 and 22, for each random switch Ξ_u. Therefore, $\forall \; \Xi_u$, either $\exists \; l \neq i(u) : \; \xi_l = 0$ (if one of the equations defined by Constraint 21 is bounded), or $\xi_{u(i)} = 0$ (i.e., Constraint 22 is bounded). In order to price the remaining free variables ξ_l, (i) we remove the variables priced to zero from the set of variables engaged by

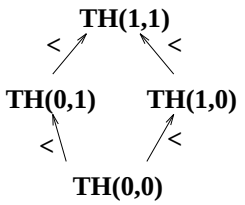


Figure 4: Example: Characterizing the dominance among the candidate scheduling policies

the original formulation of Equations 15, 13 and 14, and furthermore, (ii) we set equal to one all variables ξ_l that belong to a random switch Ξ_u which constitutes a singleton (set) after the variable elimination of Step (i). The resulting formulation preserves the structure of the original one of Equations 15, 13 and 14, but it engages a reduced set of variables. Hence, the truth of Theorem 1, is established by repetitively applying the entire argument developed above on this reduced formulation and all the subsequent formulations derived from it, while taking into consideration the finiteness of the initial sets Ξ_u. $\diamondsuit$

We notice that a solution of the type defined in Theorem 1, corresponding to a deterministic scheduling policy for the underlying GSPN, constitutes an *extreme point* [11] for the polyhedron defined by Equations 13 and 14. The next example demonstrates how the result of Theorem 1 facilitates the computation of an optimal scheduling policy for any given instance from the considered GSPN class, through an enumerative approach that terminates in a finite number of steps.

Example Theorem 1 implies that an optimal scheduling policy for the modified EMC of Figure 3 can be obtained by (i) computing, through Equations 10 – 12, the closed-form expressions for $TH(0,0)$, $TH(0,1)$, $TH(1,0)$ and $TH(1,1)$, and (ii) determining the parameter ranges over which each of these expressions dominates the others. Working according to this plan, one can establish that the dominance relationships among these four expressions are those depicted by the lattice of Figure 4. $\diamondsuit$

The reader can verify that the optimal policy, defined by $(\xi_1 = 1, \xi_2 = 1)$, essentially

implements the *First-Buffer-First-Serve (FBFS)* [30] policy on the re-entrant line of Figure 1. On the other hand, the *Last-Buffer-First-Serve (LBFS)* [30] policy corresponds to the deterministic scheduling policy defined by $(\xi_1 = 1, \xi_2 = 0)$, and as it is shown in Figure 4, it is a suboptimal policy. This result is drastically different from the situation applying to the original model of uncapacitated re-entrant lines, where the LBFS policy has been shown to be optimal – i.e., it maximizes the long-run system throughput – over all possible configurations [30]. Hence, this example and the overall analysis pursued in this work corroborate the findings of the work presented in [41], and establish the fundamental difference between the structure of the optimal scheduling policies in capacitated and uncapacitated re-entrant lines, under a stochastic operational regime which is broader than the deterministic case considered in [41].

Concluding this section, we notice that the result of Theorem 1, regarding the existence of a deterministic optimal scheduling policy, can be immediately generalized to any other MP formulation obtained from that of Equations 10 – 14, by replacing Equation 10 by any other weighted sum of the steady-state probabilities π_k. Such an objective can be, for instance, the minimization of the average Work-In-Process, $\overline{WIP}$, of the re-entrant line under steady-state operation, defined by:

$$\overline{WIP} = \sum_k \pi_k \cdot WIP(s_k), \tag{23}$$

where $WIP(s_k)$ denotes the number of parts loaded in the system in state s_k. In fact, the result of Theorem 1 applies also to the objective of minimizing the job average sojourn time, $\overline{\tau}$, since (i) by Little's law, this quantity can be expressed by

$$\overline{\tau} = \frac{\overline{WIP}}{TH} \tag{24}$$

and (ii) Equations 10 and 23 imply that, when expressed as fractional functions of ξ, both quantities $\overline{WIP}$ and TH have the same denominator $D(\xi)$ defined in Equation 19.

CHAPTER 4

MARKOV DECISION PROCESS-BASED PERFORMANCE ANALYSIS AND CONTROL OF CAPACITATED RE-ENTRANT LINES

While the GSPN-based framework proposed in the previous chapter can support a systematic and profound understanding of the system dynamics for addressing the scheduling problem in capacitated re-entrant lines, the computational complexity arising from enumerating all the state space constitutes a severe limitation in applying the approach to any practical production environment. This chapter will utilize the GSPN-based results towards the development of a systematic and computationally effective method for computing the optimal scheduling policy by (i) transforming the scheduling problem, defined in the previous chapter, to a Markov Decision Process (MDP) problem, and (ii) developing an algorithm that systematically generates the MDP formulation for any given fab configuration, while leading to a substantial reduction of the underlying state space. Even though this approach remains too complex for practical implementation, it provides further qualitative insights and a benchmarking baseline for the design of novel and efficient approximating schemes, that are presented in the next chapter.

4.1 The Continuous-Time Markov Decision Process

4.1.1 The Semi-Markov Decision Process

The previous chapter presented a detailed characterization of the untimed and timed dynamics of the capacitated re-entrant line, as manifested in the GSPN modeling framework. Specifically, if a system is at a tangible state, the corresponding transition probability distribution over all the enabled timed transitions constitutes an "*exponential race*" among these events. In the particular case that a tangible state has only one timed transition, this transition will fire *spontaneously* after some time with probability one. However, if a system

is at a vanishing state with more than two immediate transitions simultaneously enabled, random switches are used to represent and resolve the conflicts among them. More specifically, random switches provide the (discrete) probability distribution according to which one of the conflicting immediate transitions will be selected for firing. Obviously, if a vanishing state has only one immediate transition, then it fires immediately with probability one. From the above discussion, it follows that random switches constitute controls that are externally imposed in order to regulate the behavior at vanishing states with conflicting choices in the *marking process* of the GSPN.

As it was established in the previous chapter, of particular interest in this work are random switches that deterministically select only one immediate transition among those enabled at each vanishing marking with conflicting choices; such random switches will be called *deterministic* in the context of our analysis. By further assuming that vanishing states with only one immediate transition have a "single-option" random switch, with value 1, we can perceive tangible states as states eventually resulting from vanishing states through the application of a sequence of decisions. Indeed, as it was discussed above, the process evolution from a tangible marking is not amenable to any external control. Therefore, by focusing on the sub-process of vanishing states, we can abstract a model, which will allow for the explicit representation of the scheduling logic required for throughput maximization; this model constitutes a *semi-Markov Decision Process* and we shall refer to it as the *decision making process* underlying the GSPN model.

Example The GSPN in Figure 2 starts its evolution by firing the immediate transition T_{1a}, transferring a new job to stage J_1, and, as mentioned earlier, the resulting untimed system dynamics are depicted in Figure 3. The modified EMC, depicted with solid lines in Figure 3, contains only two random switches of two options each at state s_{10} and state s_{14}, providing the probability distribution according to which one of the conflicting immediate transitions will be selected for firing. The remaining vanishing states contain only one immediate transition that fires immediately, once it is enabled, which allows us to logically assign random switches with value 1 to them. Hence, the marking process can be considered

to be a decision making process underlying the GSPN model with (*discrete*) decision epochs at vanishing states. The net markings corresponding to the various states are listed in Table 1; there are 70 states, consisting of 23 tangible states and 47 vanishing states. $\diamondsuit$

4.1.2 Model Simplification: The assumption of zero-transfer times

In the subsequent discussion, it is further assumed that the transfer times are negligible with respect to the processing times involved[1]; this can be modeled by letting the rates of the exponential distribution associated with the transfer times go to infinity. From a practical standpoint, this assumption implies that the system transporter is available at any time, and leads to some effective aggregation of the underlying EMC space. A first reason for this aggregation comes from the fact that some state components required to monitor the allocation of the system transporter now become redundant (irrelevant). Moreover, a job in transit, under the original representation, can now be considered to be in the next job stage waiting for processing, or unloaded to the I/O station, in case of being in transit from the last stage J_M. This effect introduces another type of state aggregation as it makes state components representing the number of jobs in transit redundant. In more technical terms, if a tangible state has a timed transition modeling the (physical) transferring of a job to its next stage, under the considered assumption, this transition becomes an untimed / immediate transition and the state becomes a vanishing state. At the same time, any other timed transitions enabled at the considered tangible state should be treated as not enabled any more, and the resulting states become inaccessible.

However, it is still necessary to establish a scheme for representing some deliberate idleness of the system transporter in the GSPN model simplified under the assumption of zero-transfer times. This effect can be modeled by introducing a control of "Do nothing".

Definition 1 (Do nothing) *A control action that makes a system resource idle is called a "Do nothing" control.*

Deliberate idleness occurs when resource-enabled and safe job transfers are negated

[1]This is a valid assumption for many highly integrated operational contexts used in contemporary fabs, like the cluster tool, where a central robotic manipulator rapidly transfers jobs among a set of process chambers allocated around it.

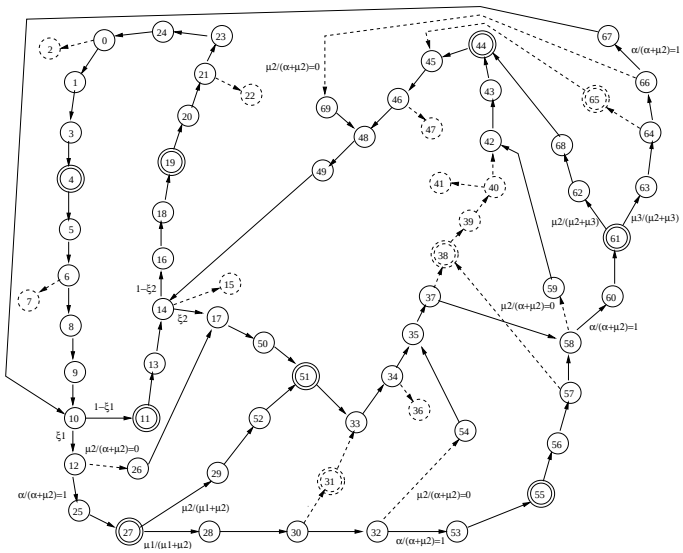


Figure 5: Example: The effect of zero-transfer times

due to performance considerations. More specifically, while deliberate idleness leading to unnecessary delay should be excluded from consideration as in the original EMC, it is still applicable when the advancement of certain jobs will introduce blocking of other jobs contesting for the same set of resources, with potential adversarial effects on the system performance.

Example The state aggregation resulting from assuming zero-transfer times for the GSPN of Figure 2 can be accomplished in two steps:

- All tangible states with an enabled timed transition corresponding to job transfer must be changed into vanishing states, while making inaccessible the remaining timed transitions at each such tangible state; the resulting EMC is depicted in Figure 5. As a concrete example, let us consider state s_{12}, which has two enabled timed transitions, respectively corresponding to transferring a job from stage J_1 to stage J_2 and processing a job in W_2. Under the assumption of zero-transfer times, the transition corresponding to the job transfer becomes an immediate one, which changes the state s_{12} into a vanishing state, while leaving state s_{26} inaccessible from s_{12}. Transitions

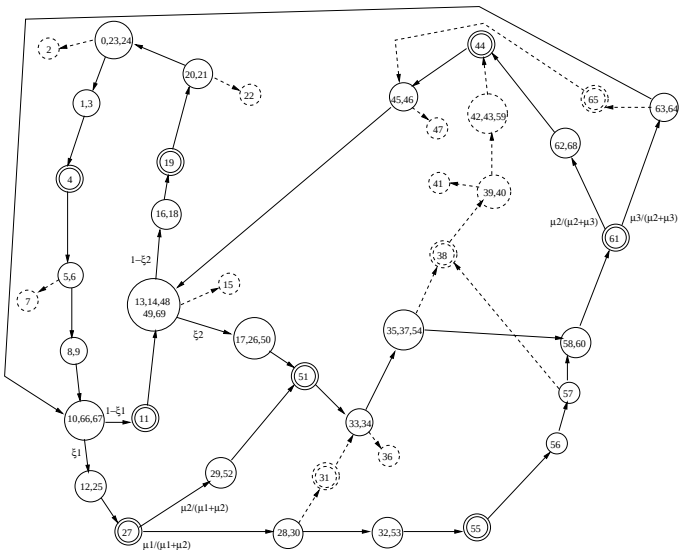


Figure 6: Example: The New EMC considering zero-transfer times

depicted in dashed lines at states s_{32}, s_{58}, and s_{66} are explained in the same way. On the other hand, state s_1, which is an example of a tangible state with only one timed transition transferring a job from the I/O station to stage J_1, simply becomes a vanishing state under the zero-transfer times assumption.

- Further state aggregation can be performed based on the facts that (i) some state components representing the allocation status of the system transporter are now becoming irrelevant for identifying states and (ii) a job in transit can be considered to be in the next stage waiting for processing. Figure 6 depicts the EMC obtained by performing the aforementioned state aggregation. Notation for states and transitions are the same as in Figure 3. However, the numbers in the circles represent the states now aggregated, and the part of the chain depicted in dashed line corresponds to transitions resulting from applying "Do nothing" controls, that should be inaccessible under operation by any optimal scheduling policy, since they essentially introduce some unnecessary delay in the system operation, as mentioned before. As an example of type (i) aggregation, state s_5 and s_6 are aggregated since they represent the same state under the considered zero-transfer times assumption: Both of them correspond

to one job being blocked in stage J_1, even though each of these two states originally represented a different allocation for the system transporter. For an example of type (ii) aggregation, state s_{12} is aggregated with state s_{25}, which is the state obtained from state s_{12} by advancing one job from stage J_1 to stage J_2.

$\diamondsuit$

Economic representation of the state Next, we provide an alternative more compact state representation for the new EMC, generated under the assumption of the zero-transfer times. This representation is suggested by the following observations: First of all, there is no need to represent explicitly the availability of the system resources such as available buffer space and available number of servers in the state representation, since this information can be inferred from other state information such as the number of jobs waiting, being processed, and being blocked at each workstation. Furthermore, the zero-transfer times assumption, when combined with the throughput maximization objective, implies that the state component representing the number of jobs blocked in the last job stage is also redundant, leading to one more step of state aggregation. More specifically, any job completed in the last stage can be considered to be unloaded immediately. Based on these arguments, we refine the definition of state as follows:

Definition 2 *Let n_{jw}, n_{jp}, and n_{jb} be the number of jobs waiting, being processed, and blocked respectively, at the job stage J_j, $j = 1, 2, \ldots, M$, where M is the number of job stages to completion. Then, under the assumption of zero-transfer times, the state of the system is defined by the (3M-1) dimensional vector $< n_{1w}, n_{1p}, n_{1b}, n_{2w}, n_{2p}, n_{2b}, \ldots, n_{Mw}, n_{Mp} >$.*

The EMC obtained under the state representation of Definition 2 has the same chain structure as the one generated by the GSPN modeling framework under the zero-transfer times assumption, however, with even more condensed state information.

Example Figure 7 shows the results from applying the economic state representation of Definition 2 to the EMC in Figure 6. As an example of the additional state aggregation

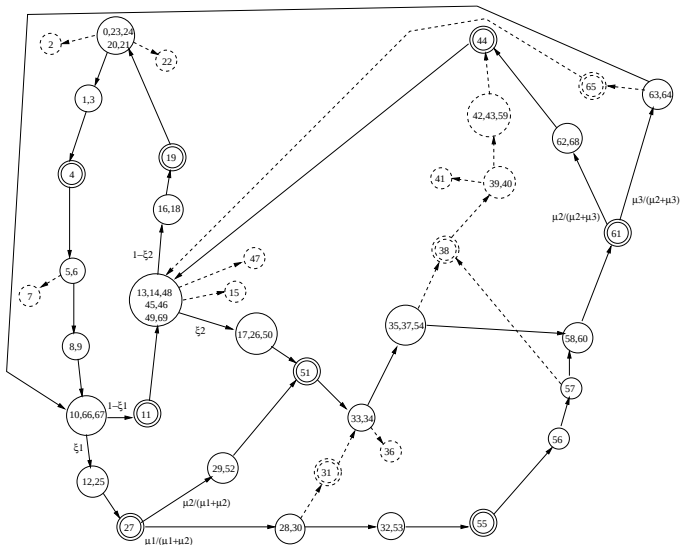


Figure 7: Example: The EMC with economic state representation

resulting from removing the state component representing the number of jobs blocked in the last job stage, state $s_{0,23,24}$ is aggregated with state $s_{20,21}$, since state $s_{20,21}$ has a job in the last stage, being completed and blocked, which can be considered to be unloaded immediately. The aggregation of state $s_{13,14,48,49,69}$ and state $s_{45,46}$ can be explained in the same way. $\diamondsuit$

4.1.3 The Continuous-Time Markov Decision Process

Clustering untimed controls An additional type of state aggregation can result from the observation that a sequence of untimed controls can be executed continuously and immediately. Hence, such a sequence of untimed controls can be clustered into a single transition, implicitly aggregating, in the process, the intermediately visited vanishing states. More specifically, a vanishing state, which results from a tangible state, can cluster untimed controls pertaining to vanishing states on the path from that state to the next tangible state; we shall call the untimed controls thus clustered a *clustered control* and the resulting tangible state a *resultant tangible state*. Under the proposed reduction, the only vanishing states appearing in the modified EMC will be those with incoming transitions from tangible states, and the number of clustered controls at such a vanishing state will be equal to

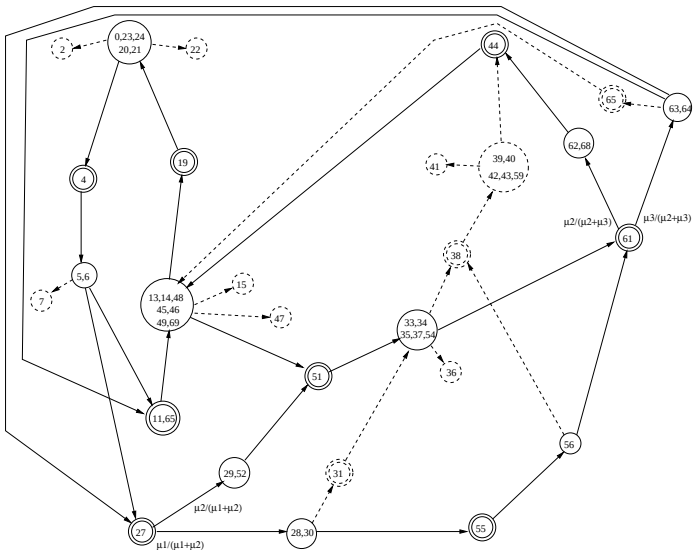


Figure 8: Example: The aggregated EMC resulting from clustering untimed controls

the number of resultant tangible states for that state. To facilitate complications in the subsequent analysis arising from an immediate succession of tangible states in the EMC, we introduce the additional convention that a state resulting from a tangible state is a vanishing state even though only timed transitions are enabled in that state. From a more technical standpoint, it is assumed that the aforementioned state is a vanishing state with only the "Do nothing" control enabled.

Example Figure 8 depicts the EMC obtained through clustering untimed controls in the EMC in Figure 7, i.e., each transition from a vanishing state to a tangible state is the result of applying a *sequence* of controls, i.e., a clustered control, while visiting a sequence of vanishing states in the EMC of Figure 7. Vanishing states that are thus implicitly aggregated to other states are not depicted in the aggregated EMC of Figure 8. $\diamondsuit$

CT-MDP Next we show that the vanishing-state process resulting from the reduction described in the previous paragraph, constitutes a *Continuous-Time Markov Decision Process (CT-MDP)*, since, in the new aggregated EMC, there exists one resultant tangible state between every pair of vanishing states. Let $\mathcal{S}_{\mathcal{V}}$ and $\mathcal{S}_{\mathcal{T}}$ respectively denote the set

of vanishing states and resultant tangible states in this aggregated EMC. 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; this set of controls corresponds to all the process and SCP-enabled event sequences that bring the system to a 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,

$$p_{ij}(u) = \frac{\sum_{l:s(l)=j} \mu_l}{\sum_{k\in\Psi(i,u)} \mu_k}. \tag{25}$$

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

$$\bar{\tau}_i(u) = \frac{1}{\sum_{k\in\Psi(i,u)} \mu_k}. \tag{26}$$

Let X_k be the system vanishing state at the k-th decision epoch t_k, and u_k be the selected control at t_k. Then, it is clear from the above discussion that $\{X_k, k = 0,1,\ldots\}$ is a *Continuous Time Markov Decision Process (CT-MDP)* with $\bar{\tau}_i(u_k) > 0$, $k = 0,1,2,\ldots$

Finally, notice that, while the chain structure of the aforementioned CT-MDP is *Multichain* [38], i.e., in general, there will exist a stationary policy with a transition matrix containing two or more closed irreducible recurrent classes, the applied structural control / deadlock avoidance policy ensures that it is also *Communicating* [38], i.e., for every pair of vanishing states, there exist deterministic stationary policies that render them accessible to each other.

4.2 An Efficient Algorithm for Generating the State Space of the Aggregated EMC

4.2.1 The State Space Generation Algorithm

Up to this point, we have described a systematic procedure for generating the state space of the aggregated EMC, starting from the marking process of the GSPN model for the scheduling problem of the capacitated re-entrant line, while further assuming zero-transfer times. The presented procedure involved generating the original state space for the GSPN model,

aggregating states by further assuming zero-transfer times, representing states economically, and clustering a sequence of untimed controls. Next we present a more efficient algorithm that generates the state space of the aggregated EMC 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.

Identifying the 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 [29]. However, in [29], 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 null state [45], or (ii) proceed to the identification of the safe region by generating and trimming with respect to the null 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 [7] for the algorithmic details.

Generating the CT-MDP state space After having obtained a characterization of the safe region, the state space consisting of $\mathcal{S}_{\mathcal{V}}$ and $\mathcal{S}_{\mathcal{T}}$ is generated systematically: For each vanishing state, all resultant tangible states are generated by enumerating all possible 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

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

1. 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 null state.

2. If all states in $\mathcal{S}_{\mathcal{V}}$ are marked as "explored", then go to Step 6. 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.

3. 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.

4. Put the resultant tangible states generated in Step 2 into $\mathcal{S}_{\mathcal{T}}$, while avoiding duplication.

5. Save the transitional information and go to Step 2.

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

4.2.2 Systematic Exploration of a Vanishing State

We begin this subsection by discussing some pertinent observations useful for the 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.

Optimality conditions of non-idling policy 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 3 *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 4 *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 5 *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 6 *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 7 *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 8 *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 9 *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 3 – 9 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 [17](Chapter 3). Here we shall provide a more intuitive justification of their correctness. Hence, Lemma 3 addresses the non-optimality of a globally idling policy for the throughput maximization problem. Lemma 4 describes the optimality of a local non-idling policy for a workstation with its buffer full of non-completed job instances. Lemma 5

represents 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 6 further generalizes Lemma 5 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 7 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 8 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 9 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.

A systematic procedure for supporting Step 2 of the state space generation algorithm can be based on a *Breadth First Search* method, where the search is further controlled by means of Lemmas 3 – 9. The developed search method takes also into consideration the requirement that in the generated EMC, clustered controls should be in one-to-one correspondence with the resultant tangible states. In other words, the presented method avoids generating redundant clustered controls that consist of different sequences of untimed controls resulting, however, in the same vanishing state; this is achieved by storing only the generated vanishing states, while ignoring the interconnecting transitions.

Breadth First Search for exploring a vanishing state

1. Given a vanishing state, mark it by "V" and let it be the root node of a search tree.

2. If all nodes marked with "V" in the generated tree are also marked with "explored", then go to Step 6. Otherwise, select one node with marking "V" in the tree, which is not explored. Let it be the POINTER node and mark it as "explored".

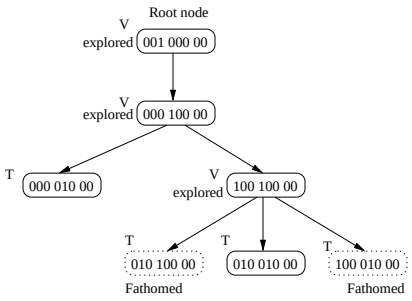


Figure 9: Example: Breadth First Search exploring a vanishing state

Table 2: Example: Vanishing states generated by the suggested algorithm

VS_k	$n_{1w}n_{1p}n_{1b}$	$n_{2w}n_{2p}n_{2b}$	$n_{3w}n_{3p}$	resultant TS_k	next VS_k
0	0 0 0	0 0 0	0 0	(TS_1)	(VS_1)
1	0 0 1	0 0 0	0 0	(TS_2,TS_3)	(VS_2), (VS_3,VS_4)
2	0 0 0	0 0 1	0 0	(TS_4,TS_5)	(VS_5), (VS_0)
3	0 0 1	0 1 0	0 0	(TS_6)	(VS_6)
4	0 1 0	0 0 1	0 0	(TS_4)	(VS_5)
5	0 0 1	0 0 1	0 0	(TS_7)	(VS_7),(VS_8)
6	0 0 0	1 0 1	0 0	(TS_7)	(VS_7),(VS_8)
7	0 0 0	0 0 1	0 1	(TS_8)	(VS_2)
8	0 0 0	0 1 0	0 0	(TS_2,TS_3)	(VS_2),(VS_3,VS_4)

3. Generation of resultant tangible states: For the POINTER node, if there is any job that can be processed or in processing, generate child nodes, with marking "T", corresponding to the operation of processing that job, while fathoming nodes violating Lemmas 3 – 9. A "Do nothing" control should also be considered in case that there is at least one job being processed.

4. One step exploration by loading or transferring operation: For the POINTER node, generate all child nodes corresponding to possible loading or transferring operations, and mark them by "V", while avoiding duplication.

5. Check if the generated nodes in Step 4 are in SR; if they are not in SR, fathom them. Go to Step 2.

6. Done with the resultant tangible states marked by "T".

Example Figure 9 shows a procedure of systematically examining all clustered controls emanating from a vanishing state $VS_1 = <$ 001 000 00 $>$, in Table 2, using the proposed

Table 3: Example: Resultant tangible states generated by the suggested algorithm

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)

Breadth First Search method. More specifically, the search begins with VS_1 as the root node, marked by "V" and "explored", which is further selected as the POINTER node in Step 2. Since there is no job that can be processed or being processed, a child node $<$ 000 100 00 $>$ corresponding to transferring a job from stage J_1 to stage J_2 is generated, in Step 4. This node is marked by "V" and is found to be in SR. Repeating Steps 2 – 5 with the node $<$ 000 100 00 $>$ as the POINTER node, two child nodes are generated; one is a resultant tangible state $<$ 000 010 00 $>$, with marking "T", corresponding to processing a job in stage J_2, and the other one is an intermediate vanishing state $<$ 100 100 00 $>$ with marking "V" within SR, representing loading a new job into stage J_1. With the newly generated POINTER node $<$ 100 100 00 $>$, one resultant tangible state $<$ 010 010 00 $>$ is generated, while two other nodes $<$ 100 010 00 $>$ and $<$ 010 100 00 $>$ are fathomed by Lemmas 4 and 5 respectively. Finally, the search ends with two nodes $<$ 000 010 00 $>$ and $<$ 010 010 00 $>$ as the resultant tangible states of the vanishing state VS_1. $\diamondsuit$

Example Figure 10 depicts the aggregated EMC generated by the proposed algorithm for the system in Figure 1, under the zero-transfer times assumption and using the optimal SCP represented by Equation 9. In Figure 10, $u_{i,k}$ represents the k-th control associated with state $VS_i \in \mathcal{S}_\mathcal{V}$; the specific action corresponding to control $u_{i,k}$ can be inferred from the corresponding vanishing state VS_i and its resultant tangible state information in Tables 2 and 3, which are equivalent to the results depicted in Figure 8. $\diamondsuit$

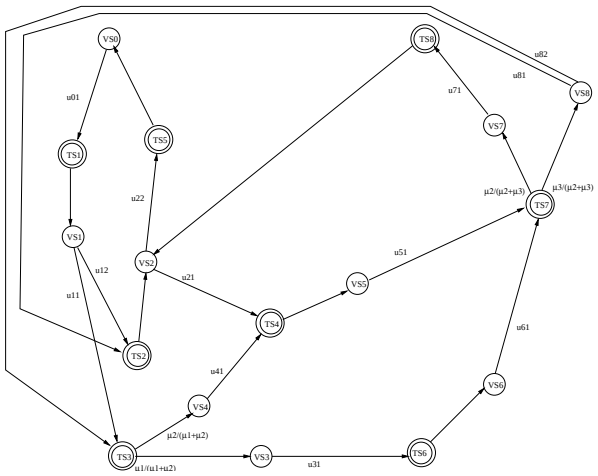


Figure 10: Example: The aggregated EMC generated by the suggested algorithm

4.3 Formulation of the Continuous-Time Average Reward MDP problem and a Solution Approach through Uniformization

4.3.1 Formulation of the CT-AR-MDP problem

The decision making process induced by $\{X_k, k = 0, 1, \ldots\}$ through the objective of maximizing the long-term / steady-state system throughput constitutes a continuous-time average reward Markov decision process (CT-AR-MDP). In the CT-AR-MDP framework, the long-run throughput of the capacitated re-entrant line is modeled by the (time-)*average reward* to be accumulated by the considered process, formally defined by

$$\lim_{N\to\infty} \frac{1}{E\{t_N\}} E\Big\{ \int_0^{t_N} g(x(t), u(t))dt \Big\}. \tag{27}$$

In equation 27, $g(x(t), u(t))$ is the *reward* per unit time obtained by taking control $u(t)$ at state $x(t)$ at time t, where $x(t) = x_k$ and $u(t) = u_k$ for $t_k \leq t < t_{k+1}$. Furthermore, the single-stage expected reward, $G(i, u)$, corresponding to state i and control u, is given by

$$G(i, u) = g(i, u)\bar{\tau}_i(u), \tag{28}$$

where $\bar{\tau}_i(u)$ is the expected value of the transition time corresponding to state-control pair (i, u). Next we show how to systematically derive the problem parameters, and proceed to its solution through the employment of well-established algorithms.

The single-stage expected reward Under the assumption of zero-transfer times, the single-stage expected reward for the fab scheduling problem of the capacitated re-entrant line can be computed by considering that a non-zero reward occurs only if the resultant tangible state corresponding to state-control pair (i, u), has a job being processed in the last job stage J_M; then, the reward per unit time, $g(i, u)$, is defined as follows:

$$g(i,u) = \begin{cases} \mu_M & \text{if a resultant tangible state has a job} \\ & \text{being processed in the last stage } J_M \\ 0 & \text{otherwise.} \end{cases} \tag{29}$$

From Equation 28, we have

$$G(i,u) = \begin{cases} \mu_M \bar{\tau}_i(u) & \text{if } g(i,u) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{30}$$

Example The reward per unit time and the single-stage expected reward for the CT-AR-MDP defined by the vanishing-state process in Figure 10 are given by

$$g(i,u_{i,k}) = \begin{cases} \mu_3 & \text{if } (i,k) = (2,2),(5,1),(6,1),(7,1) \\ 0 & \text{otherwise,} \end{cases} \tag{31}$$

and

$$G(i,u_{i,k}) = \begin{cases} 1 & \text{if } (i,k) = (2,2),(7,1) \\ \frac{\mu_3}{\mu_2+\mu_3} & \text{if } (i,k) = (5,1),(6,1) \\ 0 & \text{otherwise,} \end{cases} \tag{32}$$

where $u_{i,k}$ represents the k-th control associated with state VS_i in Figure 10. $\diamondsuit$

Formulation of Bellman's optimality equation 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 [38], and furthermore, there exists a function $h^*(i)$, $i \in \mathcal{S}_\mathcal{V}$, that satisfies the following equation, for all states $i \in \mathcal{S}_\mathcal{V}$,

$$h^*(i) = \max_{u \in U(i)} \left[G(i,u) - \lambda^* \bar{\tau}_i(u) + \sum_{j \in \mathcal{S}_\mathcal{V}} p_{ij}(u) h^*(j) \right] \tag{33}$$

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

$$\mu^*(i) = \arg\max_{u \in U(i)} \Big[G(i,u) - \lambda^* \bar{\tau}_i(u) + \sum_{j \in \mathcal{S}_\mathcal{V}} p_{ij}(u) h^*(j) \Big] \tag{34}$$

4.3.2 Uniformization

Based on the MDP theory [4, 38], the CT-AR-MDP problem can be analyzed more easily by converting it to an equivalent problem through uniformization, a process that observes the state of the underlying system at random times, which are exponentially distributed, but with uniform sojourn time distribution in every state and with an observation rate much faster than the rates of the occurring events. More specifically, let γ be any scalar defining the mean of the exponentially distributed time intervals for observing the state of the system, such that

$$0 < \gamma < \frac{\bar{\tau}_i(u)}{1 - p_{ii}(u)}, \tag{35}$$

for all states $i \in \mathcal{S}_\mathcal{V}$ and controls $u \in U(i)$ [4, 38]. Also, let $\{\widetilde{X}_k, k = 0, 1, \ldots\}$ denote the new state process resulting from uniformization, i.e., $\tilde{x}_k$ is the state of the system at time t_k, the time of k-th observation. Then, since, in our case, $p_{ii} = 0$ for all states $i \in \mathcal{S}_\mathcal{V}$, Equation 35 becomes

$$0 < \gamma < \bar{\tau}_i(u), \tag{36}$$

for all states $i \in \mathcal{S}_\mathcal{V}$ and $u \in U(i)$. By defining, for all states $i, j \in \mathcal{S}_\mathcal{V}$ and $u \in U(i)$,

$$\tilde{p}_{ij}(u) = \begin{cases} \frac{\gamma}{\bar{\tau}_i(u)} p_{ij}(u) & \text{if } j \neq i \\ 1 - \frac{\gamma}{\bar{\tau}_i(u)} & \text{if } j = i, \end{cases} \tag{37}$$

$\tilde{p}_{ij}(u)$ become the transition probabilities of a discrete-time average reward MDP problem [4], and $\{\widetilde{X}_k, k = 0, 1, \ldots\}$ constitutes a discrete-time AR-MDP (DT-AR-MDP) with the same state space $\mathcal{S}_\mathcal{V}$ and the same infinitesimal generator as the original CT-AR-MDP, i.e., the original and the new process are equal *in distribution* [38]. Furthermore, the specific selection of γ satisfying Equation 36 does not affect the characteristics of the generated state processes, i.e., all these state processes generated using different values of γ

for uniformization are equivalent in distribution and have equivalent Bellman's optimality equation, eventually leading to the same optimal policy.

4.3.3 Bellman's Optimality Equation for the DT-AR-MDP

Bellman's optimality equation for the process $\{\widetilde{X}_k, k = 0, 1, \ldots\}$ can be derived from Equation 33, so that it reflects the rescaling of time introduced by uniformization, as follows: For all $i \in \mathcal{S}_{\mathcal{V}}$, Equation 33 is equivalent to

$$h^*(i) \geq G(i,u) - \lambda^* \bar{\tau}_i(u) + \sum_{j \in \mathcal{S}_{\mathcal{V}}} p_{ij}(u) h^*(j), u \in U(i) \tag{38}$$

with at least one equality constraint corresponding to an optimal control at each state i. Then, by multiplying by $\frac{\gamma}{\bar{\tau}_i(u)}$ both sides of Equation 38 and using Equation 28, we have, for all $i \in \mathcal{S}_{\mathcal{V}}$,

$$\frac{\gamma}{\bar{\tau}_i(u)} h^*(i) \geq \gamma g(i,u) - \gamma\lambda^* + \sum_{j \in \mathcal{S}_{\mathcal{V}}} \frac{\gamma}{\bar{\tau}_i(u)} p_{ij}(u) h^*(j), u \in U(i) \tag{39}$$

By using Equation 37, this is equivalent to, for all $i \in \mathcal{S}_{\mathcal{V}}$,

$$h^*(i) \geq \gamma g(i,u) - \gamma\lambda^* + \sum_{j \in \mathcal{S}_{\mathcal{V}},\ j \neq i} \tilde{p}_{ij}(u) h^*(j) + (1 - \frac{\gamma}{\bar{\tau}_i(u)}) h^*(i), u \in U(i), \tag{40}$$

which can be simplified to, for all $i \in \mathcal{S}_{\mathcal{V}}$,

$$\frac{h^*(i)}{\gamma} \geq g(i,u) - \lambda^* + \sum_{j \in \mathcal{S}_{\mathcal{V}}} \tilde{p}_{ij}(u) \frac{h^*(j)}{\gamma}, u \in U(i), \tag{41}$$

and finally, we have

$$\frac{h^*(i)}{\gamma} = \max_{u \in U(i)} \left[g(i,u) - \lambda^* + \sum_{j \in \mathcal{S}_{\mathcal{V}}} \tilde{p}_{ij}(u) \frac{h^*(j)}{\gamma} \right], \forall\ i \in \mathcal{S}_{\mathcal{V}}, \tag{42}$$

which is equivalent to

$$\tilde{h}^*(i) = \max_{u \in U(i)} \left[g(i,u) - \lambda^* + \sum_{j \in \mathcal{S}_{\mathcal{V}}} \tilde{p}_{ij}(u) \tilde{h}^*(j) \right], \forall\ i \in \mathcal{S}_{\mathcal{V}}, \tag{43}$$

where $\tilde{h}^*(i) = \frac{h^*(i)}{\gamma}$.

Equation 43 is Bellman's optimality equation for the CT-AR-MDP corresponding to a unit time interval, and it can be interpreted as the Bellman's optimality equation for

the DT-AR-MDP with immediate reward $g(i,u)$ for state-control pair (i,u) and optimal average reward per stage, λ^*. The corresponding optimal policy is defined as follows:

$$\mu^*(i) = \arg\max_{u \in U(i)} \left[g(i,u) + \sum_{j \in \mathcal{S}_\mathcal{V}} \tilde{p}_{ij}(u) \tilde{h}^*(j) \right] \tag{44}$$

From a computational standpoint, the construction of Equation 43 can be facilitated by the following lemma:

Lemma 10 *Let $q_{ij}(u)$ be the transition rate from state i to state j resulting from state-control pair (i,u) in the CT-AR-MDP. Then, for states $i, j \in \mathcal{S}_\mathcal{V}$, and $u \in U(i)$,*

$$\tilde{p}_{ij}(u) = \begin{cases} \gamma q_{ij}(u) & \text{for } j \neq i \\ 1 - \gamma \sum_{k \neq i} q_{ik}(u) & \text{for } j = i. \end{cases} \tag{45}$$

Proof: By definition,

$$q_{ij}(u) = p_{ij}(u) \frac{1}{\bar{\tau}_i(u)}. \tag{46}$$

Then, for $j \neq i$, from Equation 37,

$$\tilde{p}_{ij}(u) = \frac{\gamma}{\bar{\tau}_i(u)} p_{ij}(u) = \frac{\gamma}{\bar{\tau}_i(u)} q_{ij}(u) \bar{\tau}_i(u) = \gamma q_{ij}(u), \tag{47}$$

and for $j = i$,

$$\tilde{p}_{ij}(u) = 1 - \frac{\gamma}{\bar{\tau}_i(u)} = 1 - \gamma \sum_{k \neq i} q_{ik}(u). \tag{48}$$

$\diamondsuit$

Example The single-stage expected reward and transition probabilities for the DT-AR-MDP, generated through uniformization from the CT-AR-MDP that is defined by the vanishing-state process in Figure 10, are computed as follows:

$$g(i, u_{i,k}) = \begin{cases} \mu_3 & \text{if } (i,k) = (2,2), (5,1), (6,1), (7,1) \\ 0 & \textit{otherwise}, \end{cases}$$

$$\tilde{p}_{0,1}(u_{0,1}) = \gamma\mu_1,\ \tilde{p}_{0,0}(u_{0,1}) = 1 - \gamma\mu_1,$$

$$\tilde{p}_{1,3}(u_{1,1}) = \gamma\mu_1,\ \tilde{p}_{1,4}(u_{1,1}) = \gamma\mu_2,\ \tilde{p}_{1,1}(u_{1,1}) = 1 - \gamma(\mu_1 + \mu_2),$$

$$\tilde{p}_{1,2}(u_{1,2}) = \gamma\mu_2,\ \tilde{p}_{1,1}(u_{1,2}) = 1 - \gamma\mu_2,$$

$$\tilde{p}_{2,5}(u_{2,1}) = \gamma\mu_1,\ \tilde{p}_{2,2}(u_{2,1}) = 1 - \gamma\mu_1,$$

$$\tilde{p}_{2,0}(u_{2,2}) = \gamma\mu_3,\ \tilde{p}_{2,2}(u_{2,2}) = 1 - \gamma\mu_3,$$

$$\tilde{p}_{3,6}(u_{3,1}) = \gamma\mu_2,\ \tilde{p}_{3,3}(u_{3,1}) = 1 - \gamma\mu_2,$$

$$\tilde{p}_{4,5}(u_{4,1}) = \gamma\mu_1,\ \tilde{p}_{4,4}(u_{4,1}) = 1 - \gamma\mu_1,$$

$$\tilde{p}_{5,7}(u_{5,1}) = \gamma\mu_2,\ \tilde{p}_{5,8}(u_{5,1}) = \gamma\mu_3,\ \tilde{p}_{5,5}(u_{5,1}) = 1 - \gamma(\mu_2 + \mu_3),$$

$$\tilde{p}_{6,7}(u_{6,1}) = \gamma\mu_2,\ \tilde{p}_{6,8}(u_{6,1}) = \gamma\mu_3,\ \tilde{p}_{6,6}(u_{6,1}) = 1 - \gamma(\mu_2 + \mu_3),$$

$$\tilde{p}_{7,2}(u_{7,1}) = \gamma\mu_3,\ \tilde{p}_{7,7}(u_{7,1}) = 1 - \gamma\mu_3,$$

$$\tilde{p}_{8,2}(u_{8,1}) = \gamma\mu_2,\ \tilde{p}_{8,8}(u_{8,1}) = 1 - \gamma\mu_2,$$

$$\tilde{p}_{8,3}(u_{8,2}) = \gamma\mu_1,\ \tilde{p}_{8,4}(u_{8,2}) = \gamma\mu_2,\ \tilde{p}_{8,8}(u_{8,2}) = 1 - \gamma(\mu_1 + \mu_2).$$

In the above expressions, $u_{i,k}$ denotes the k-th control associated with state VS_i in Figure 10.

$\diamondsuit$

4.3.4 The Linear Programming Approach

There are several exact solution methods to solve the resulting DT-AR-MDP problem, including Value and Policy Iteration, Linear Programming and a number of variations of these basic methods. Among them, Linear Programming is a quite efficient approach if the state space and the control space are reasonably sized. Then, according to [4], the *primal* LP formulation for the considered DT-AR-MDP problem is as follows:

$$\min\ \lambda \tag{49}$$

s.t.

$$\lambda + \tilde{h}(i) \geq g(i,u) + \sum_{j \in \mathcal{S}_\mathcal{V}} \tilde{p}_{ij}(u)\tilde{h}(j), \forall\ i \in \mathcal{S}_\mathcal{V}, u \in U(i). \tag{50}$$

We notice that (λ^*, $\tilde{h}^*$) employed in Equation 43 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 49 and 50, 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 49 and 50 in order to obtain an *optimal* relative value function and its corresponding policy. This dual LP is formulated as follows:

$$\max \sum_{i \in \mathcal{S}_\mathcal{V}} \sum_{u \in U(i)} g(i,u)x(i,u) \tag{51}$$

s.t.

$$\sum_{u \in U(i)} x(i,u) - \sum_{j \in \mathcal{S}_\mathcal{V}} \sum_{u \in U(j)} \tilde{p}_{ij}(u)x(j,u) = 0, \forall\ i \in \mathcal{S}_\mathcal{V} \tag{52}$$

$$\sum_{i \in \mathcal{S}_\mathcal{V}} \sum_{u \in U(i)} x(i,u) = 1 \tag{53}$$

$$x(i,u) \geq 0, \forall\ i \in \mathcal{S}_\mathcal{V}, u \in U(i) \tag{54}$$

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

$$\frac{x^*(i,u)}{\sum_{u \in U(i)} x^*(i,u)}. \tag{55}$$

Since in Chapter 3 the optimal control policy was shown to be deterministic, 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 55 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)

The optimal relative value function $\tilde{h}^*(i)$ for this optimal policy can be computed by solving the following system of linear equations:[2]

$$\lambda + \tilde{h}(i) = g(i, \mu^*(i)) + \sum_{j \in \mathcal{S}_\mathcal{V}} \tilde{p}_{ij}(\mu^*(i))\tilde{h}(j), \forall\ i \in \mathcal{S}_\mathcal{V}. \tag{56}$$

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

Example The dual LP formulation for the DT-AR-MDP problem in the previous example is as follows:

$$\max \ \mu_3\Big[x(2,u_{2,2}) + x(5,u_{5,1}) + x(6,u_{6,1}) + x(7,u_{7,1})\Big] \tag{57}$$

s.t.

$$\begin{aligned}
\gamma\mu_1 x(0,u_{0,1}) - \gamma\mu_3 x(2,u_{2,2}) &= 0\\
\gamma(\mu_1+\mu_2)x(1,u_{1,1}) + \gamma\mu_2 x(1,u_{1,2}) - \gamma\mu_1 x(1,u_{0,1}) &= 0\\
\gamma\mu_1 x(2,u_{2,1}) + \gamma\mu_3 x(2,u_{2,2}) - \gamma\mu_2 x(1,u_{1,2}) - \gamma\mu_3 x(7,u_{7,1}) - \gamma\mu_2 x(8,u_{8,1}) &= 0\\
\gamma\mu_2 x(3,u_{3,1}) - \gamma\mu_1 x(1,u_{1,1}) - \gamma\mu_1 x(8,u_{8,2}) &= 0\\
\gamma\mu_1 x(4,u_{4,1}) - \gamma\mu_2 x(1,u_{1,1}) - \gamma\mu_2 x(8,u_{8,2}) &= 0\\
\gamma(\mu_2+\mu_3)x(5,u_{5,1}) - \gamma\mu_1 x(2,u_{2,1}) - \gamma\mu_1 x(4,u_{4,1}) &= 0\\
\gamma(\mu_2+\mu_3)x(6,u_{6,1}) - \gamma\mu_2 x(3,u_{3,1}) &= 0\\
\gamma\mu_3 x(7,u_{7,1}) - \gamma\mu_2 x(5,u_{5,1}) - \gamma\mu_2 x(6,u_{6,1}) &= 0\\
\gamma\mu_2 x(8,u_{8,1}) + \gamma(\mu_1+\mu_2)x(8,u_{8,2}) - \gamma\mu_3 x(5,u_{5,1}) - \gamma\mu_3 x(6,u_{6,1}) &= 0\\
x(0,u_{0,1}) + x(1,u_{1,1}) + x(1,u_{1,2}) + x(2,u_{2,1}) + x(2,u_{2,2}) + x(3,u_{3,1}) & \\
+x(4,u_{4,1}) + x(5,u_{5,1}) + x(6,u_{6,1}) + x(7,u_{7,1}) + x(8,u_{8,1}) + x(8,u_{8,2}) &= 1\\
\forall\ i \in \mathcal{S}_{\mathcal{V}}, u \in U(i), \quad x(i,u) &\geq 0
\end{aligned} \tag{58}$$

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

$$\mu(VS_i) = \begin{cases} u_{1,1} \text{ or } u_{1,2} & \text{for } i = 1\\ u_{2,1} & \text{for } i = 2\\ u_{8,2} & \text{for } i = 8\\ u_{i,1} & \text{otherwise,} \end{cases} \tag{59}$$

where $u_{i,k}$ be the k-th control associated with state VS_i in Figure 10. 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 what state VS_1 is a transient state. It is interesting to note 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. $\diamondsuit$

The computational capability provided by the MDP-based modeling framework developed in this chapter can be used for the characterization of the optimal scheduling policy in a few small "prototypical" fab configurations; the study of these solutions can provide the qualitative insights and the benchmarking baseline for the subsequent development of scalable approximating scheduling methods based on the emerging Neuro-Dynamic Programming (NDP) theory. This is the topic of the next chapter.

CHAPTER 5

AN EXPERIMENTAL INVESTIGATION OF FEATURE-BASED RELATIVE VALUE FUNCTION APPROXIMATION FOR PERFORMANCE CONTROL OF CAPACITATED RE-ENTRANT LINES

In the previous chapters, two approaches for addressing performance modeling, analysis and control of capacitated re-entrant lines were developed. These approaches provide the analytical basis for addressing the re-entrant line scheduling problem in the more complex operational context considered in this work, but they have a severe computational limitation in that they require the explicit enumeration of the underlying state space, which explodes very fast. In addition, it is a well-established result that the derivation of the optimal scheduling policy in the considered problem context is an NP-hard problem [15]. Therefore, there is a need for some near-optimal approximating scheme that maintains computational tractability.

5.1 Neuro-Dynamic Programming Approach

The observation of Chapter 4 that the optimal control policy for a discrete-time average reward 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 a parameterized function, according to the emerging theory of *Neuro-Dynamic Programming* (NDP), discussed in Section 2.3. The first step in the development of such an approximate representation of the optimal relative value function is to choose an *approximation architecture*, i.e., a functional form parameterized with a number of free variables, that will define the space/set of functions to be considered as candidates for the

approximation. Once the approximation architecture is selected, the approximating procedure essentially constitutes a "*tuning*" of the aforementioned free variables in order to provide a "*best fit*" of the function to be approximated. Hence, in this approach, two important issues must be considered: (i) The approximation architecture must be rich enough in order to provide a close approximation of the target – in our case, the optimal relative value – function. (ii) Effective algorithms for tuning the parameters of the approximation architecture must be available. These two issues are often conflicting. Typically, a very rich approximation architecture will involve a large set of components interrelated through some non-linear function. Both of these elements will increase the computational complexity of the tuning process and will impair the analytical study of any proposed tuning algorithm. As a result, the scientific community has currently confined itself primarily in the study of *linear* approximation architectures, i.e., architectures in which the approximating function is expressed as a *weighted sum* of some preselected *feature functions*. These feature functions must capture important aspects of the system state, and their selection must be driven by practical experience, insight and/or any formal results available for the considered problem.

Motivated by these general remarks, this research program will seek primarily to investigate the ability of the aforementioned linear architecture to provide effective approximations for the optimal relative value function of the MDP problem formulated in Chapter 4, while identifying a *"good"* set of feature functions for the considered capacitated re-entrant line scheduling problem. We proceed to the investigation of this question by formalizing first the considered approximation framework. Let Φ be a *feature space*, i.e., a set $\Phi = (\phi_0, \ldots, \phi_K)$ of functions polynomially evaluated for any given state, where $\phi_j = (\phi_j(0), \ldots, \phi_j(n-1))^T, j = 0, \ldots, K$, with $|\Phi| = K+1$, $|\mathcal{S}_\mathcal{V}| = n$, and $\phi_0(i) = 1$ for all $i \in \mathcal{S}_\mathcal{V}$. Let $r = (r_0, \ldots, r_K)$ be a parameter vector known as *weights* of the linear architecture in approximating the optimal relative value functions. Then, a feature-based linearly parameterized approximation function can be represented as follows:

$$\hat{h}(i, r) = \sum_{k=0}^{K} \phi_k(i) r_k = (\Phi r)(i). \tag{60}$$

A value of r^* satisfying $\tilde{h}^*(i) = \hat{h}(i, r^*) = (\Phi r^*)(i)$ would give us the optimal relative value

function and the corresponding *"greedy"* policy based on Equation 44 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 44 tends to maximize throughput for the underlying DT-AR-MDP problem.

In the literature [46, 51, 54, 53, 52, 18], there have been many research results about the approximation of optimal value functions using a polynomial linear approximation function as in Equation 60. Most of these works, however, focus on the problem of weight selection and are based on the assumption that a "good" set of feature functions is given; only [54, 53] give some guidelines for selecting feature functions for some discounted cost problems. As mentioned in Section 2.3, the feature function selection problem is application specific and should be investigated on a case by case basis. Our effort to identify a "good" set of feature functions for the capacitated re-entrant line scheduling problem is based on an experimental procedure that takes advantage of (i) the analytical insights available for the uncapacitated re-entrant line scheduling problem and (ii) the computational capability of generating optimal relative value functions and policies for small-sized system configurations, developed in Chapter 4. More specifically, the adopted procedure is as follows:

(i) Extract a set Φ of feature functions based on queueing theoretic concepts and results.

(ii) Generate the optimal relative value function and policy for a number of small-sized capacitated re-entrant lines with different configurations.

(iii) Fit the linear architecture defined by the selected feature set Φ of Step (i) to the optimal relative value function generated in Step (ii).

(iv) Assess the quality of the approximations and of the greedy policy provided by them.

(v) Eventually, identify a "*good*" set of feature functions that seems to provide good quality of approximations and enhanced performance, while applying the procedure for different sets of feature functions.

Next, we elaborate on the mathematical apparatus and the procedures adopted to support items (ii) – (iv). Items (i) and (v) are the topics of the Sections 5.3 and 5.4.

5.2 Mathematical Programming Formulations for Computing r^* and Evaluating the Approximating Capability of Feature Set Φ

Approximating optimal relative value functions using a feature-based max-norm projection The quality of the approximation provided by a feature set Φ can be quantified through the employment of some performance/distance metrics representing the "*goodness-of-fit*", such as the l_∞- or l_2-norm; the resulting measures are as follows:

$$\min_r \max_i \left| \tilde{h}^*(i) - \hat{h}(i,r) \right| \tag{61}$$

and

$$\min_r \sum_i \left[\tilde{h}^*(i) - \hat{h}(i,r) \right]^2. \tag{62}$$

The basic idea behind Equations 61 and 62 is to characterize the "goodness-of-fit" of the approximation by minimizing an error index between the actual optimal relative value function and its projection to the subspace spanned by Φ. More specifically, the l_∞-norm approximation tries to minimize the distance between the optimal relative value function and its approximated value as uniformly as possible over all states, whereas the l_2-norm approximation attempts to minimize the Euclidean distance between the optimal relative value function and its approximated value over all states. From a computational standpoint, the dependency of the l_∞-norm on r is piecewise linear, which might lead to the existence of alternative optimal solutions for r^*. Given $\tilde{h}^*(i)$, the mathematical formulation of Equation 61, based on the l_∞-norm, can be transformed into an LP and solved by some LP method. On the other hand, the l_2-norm constitutes a smoothing function and therefore, the optimization and analysis of the formulation of Equation 62 can be performed through derivative-based techniques. In the following, we focus on the l_∞-norm since the objective value of Equation 61 admits a more straightforward intuitive interpretation. Then, a key issue addressed in this part of the research is the extent to which a small uniform approximation error preserves the shape of the optimal relative value function under consideration.

A mathematical programming formulation for computing r^* By letting

$$\epsilon = \max_i |\tilde{h}^*(i) - \hat{h}(i,r)|, \tag{63}$$

the feature-based l_∞-norm approximation problem defined by Equation 61 can be transformed to the following nonlinear optimization problem:

$$\min\ \epsilon \tag{64}$$

s.t.

$$\left|\tilde{h}^*(i) - \hat{h}(i,r)\right| \leq \epsilon, \forall\ i \in \mathcal{S}_\mathcal{V}. \tag{65}$$

The optimal relative value function $\tilde{h}^*(i)$ utilized in this formulation can be obtained by solving the system of Equation 56. However, since Equation 56 does not have a unique optimal solution[38, 4], the optimization problem defined by Equations 64 and 65 can be transformed to an equivalent optimization problem of the form:

$$\min\ \epsilon \tag{66}$$

s.t.

$$\begin{aligned} \left|\tilde{h}(i) - \hat{h}(i,r)\right| &\leq \epsilon, \forall\ i \in \mathcal{S}_\mathcal{V} & (67)\\ \lambda + \tilde{h}(i) - \sum_{j \in \mathcal{S}_\mathcal{V}} \tilde{p}_{ij}(\mu^*(i))\tilde{h}(j) &= g(i,\mu^*(i)), \forall\ i \in \mathcal{S}_\mathcal{V}, & (68) \end{aligned}$$

which can be linearized, using Equation 60, as follows:

$$\min\ \epsilon \tag{69}$$

s.t.

$$\begin{aligned} \epsilon + \tilde{h}(i) - \sum_{k=0}^{K} \phi_k(i) r_k &\geq 0, \forall\ i \in \mathcal{S}_\mathcal{V} & (70)\\ \epsilon - \tilde{h}(i) + \sum_{k=0}^{K} \phi_k(i) r_k &\geq 0, \forall\ i \in \mathcal{S}_\mathcal{V} & (71)\\ \lambda + \tilde{h}(i) - \sum_{j \in \mathcal{S}_\mathcal{V}} \tilde{p}_{ij}(\mu^*(i))\tilde{h}(j) &= g(i,\mu^*(i)), \forall\ i \in \mathcal{S}_\mathcal{V}. & (72) \end{aligned}$$

An optimal solution r^* to this MP problem minimizes the l_∞-norm-based projection error of $\tilde{h}^*(i)$ to the subspace spanned by Φ and the corresponding optimal objective value ϵ^* represents the quality of the approximation achieved by a set Φ of feature functions.

Obtaining a greedy policy and a greedy throughput Let r^* be an optimal solution to the MP formulation of Equations 69 – 72. Then, a greedy policy determined by r^* i.e., by an approximation of the optimal relative value function according to $\hat{h}(i, r^*) = \sum_{k=0}^{K} \phi_k(i) r_k^*$ can be obtained by

$$\mu(i, r^*) = \arg \max_{u \in U(i)} \Big[g(i, u) + \sum_{j \in \mathcal{S}_\mathcal{V}} \tilde{p}_{ij}(u) \hat{h}(j, r^*) \Big], \forall\ i \in \mathcal{S}_\mathcal{V}. \tag{73}$$

The throughput resulting from the greedy policy is the *ultimate measure* for the quality of the feature space Φ.

The following lemma is a rather technical result and it shows that the proposed evaluation scheme for a feature space Φ is not affected by the nondeterminism of the optimal relative value function indicated in footnote 2 in Section 2.3.

Lemma 11 *Let $\tilde{h}_1^*$ be an optimal solution to the MP formulation of Equations 49 and 50, and r_1^* be the corresponding optimal parameter vector to the MP formulation of Equations 69 – 72. Furthermore, let ϵ^* denote the corresponding optimal objective value to the MP formulation of Equations 69 – 72, and $\mu(i, r_1^*), \forall\ i \in \mathcal{S}_\mathcal{V}$ denote the greedy policy generated by Equation 73 based on r_1^*. Then, for any alternative optimal solution $\tilde{h}_2^*$ to the MP formulation of Equations 49 and 50 such that $\tilde{h}_2^* = \tilde{h}_1^* + ce, c \in R$ and e is the vector with all its components equal to one, the following statements hold true:*

(i) The MP formulation of Equations 69 – 72 obtained from $\tilde{h}_2^*$ has the same optimal objective value ϵ^*.

(ii) r_1^* induces an optimal solution r_2^* for the formulation of statement (i) above.

(iii) r_1^* and r_2^* generate the same greedy policy based on Equation 73 and the corresponding approximations of the optimal relative value function.

Proof:

(i) Suppose that the MP formulation of Equations 69 – 72 obtained from $\tilde{h}_2^*$ has the optimal objective value $\epsilon^{'}$ such that $\epsilon^{'} < \epsilon^*$. Then, by the optimality of $\epsilon^{'}$, there exists

r_2^* such that

$$\epsilon^{'} = \max_i \left| \tilde{h}_2^*(i) - \hat{h}(i, r_2^*) \right| \tag{74}$$

$$= \max_i \left| (\tilde{h}_1^*(i) + c) - \hat{h}(i, r_2^*) \right| \tag{75}$$

$$= \max_i \left| \tilde{h}_1^*(i) - (\hat{h}(i, r_2^*) - c) \right|. \tag{76}$$

But then, Equation 76 implies that the weight vector $r_2^* - ce_1$ provides a better-fit approximation for $\tilde{h}_1^*$ than r_1^*, where e_1 is the vector with only its first component equal to one and all other components equal to zero; this is a contradiction. A symmetrical argument establishes a similar contradiction for the case $\epsilon^{'} > \epsilon^*$. Therefore, $\epsilon^{'} = \epsilon^*$.

(ii) The proof of statement (i) establishes immediately that the sought optimal solution is: $r_2^* = r_1^* + ce_1$.

(iii) The greedy policy generated by r_2^* i.e., by an approximation of the optimal relative value function $\hat{h}(r_2^*)$ at state i is defined as follows:

$$\mu(i, r_2^*) = \arg \max_{u \in U(i)} \left[g(i,u) + \sum_{j \in \mathcal{S}_\mathcal{V}} \tilde{p}_{ij}(u) \hat{h}(j, r_2^*) \right], \forall\ i \in \mathcal{S}_\mathcal{V}, \tag{77}$$

which is equivalent to

$$\mu(i, r_2^*) = \arg \max_{u \in U(i)} \left[g(i,u) + \sum_{j \in \mathcal{S}_\mathcal{V}} \tilde{p}_{ij}(u) (\hat{h}(j, r_1^*) + c) \right], \forall\ i \in \mathcal{S}_\mathcal{V}. \tag{78}$$

Then, we have

$$\mu(i, r_2^*) = \arg \max_{u \in U(i)} \left[g(i,u) + \sum_{j \in \mathcal{S}_\mathcal{V}} \tilde{p}_{ij}(u) \hat{h}(j, r_1^*) + c \right], \forall\ i \in \mathcal{S}_\mathcal{V} \tag{79}$$

$$= \arg \max_{u \in U(i)} \left[g(i,u) + \sum_{j \in \mathcal{S}_\mathcal{V}} \tilde{p}_{ij}(u) \hat{h}(j, r_1^*) \right], \forall\ i \in \mathcal{S}_\mathcal{V} \tag{80}$$

$$= \mu(i, r_1^*), \forall\ i \in \mathcal{S}_\mathcal{V}. \tag{81}$$

$\diamondsuit$

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 greedy policy generated by the approximation, with the optimal

throughput, λ^*, and also, the throughput that would be obtained if some other heuristics were applied. We notice, however, the following additional issues that complicate and, to some extent, compromise the implementation of the proposed evaluation scheme:

- Some of the involved computations present numerical instability and the accrued errors should be filtered out to the extent possible.

- The MP formulation of Equations 64 and 65 might have alternative optimal solutions r^*, resulting in different greedy policies with different greedy 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 the Equation 43 with $\tilde{h}_1^* \neq \tilde{h}_2^* + ce$, which result in alternative optimal solutions to the MP formulation of Equations 64 and 65, leading to different greedy policies and different throughput.

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 a strictly greedy scheme. In addition, we recognize that in the eventual implementation of the proposed approximating framework, the adopted "greedy" 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. This randomization effect should be accounted for when assessing the performance resulting from the proposed approach. On the positive side, this randomizing effect restores the "unichain" 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:

$$\forall\ i \in \mathcal{S}_{\mathcal{V}},\ u \in U(i), \quad R(i,u) = g(i,u) + \sum_{j \in \mathcal{S}_{\mathcal{V}}} \tilde{p}_{ij}(u)\hat{h}(j, r^*) \tag{82}$$

$$\forall\ i \in \mathcal{S}_{\mathcal{V}},\ \widetilde{U}(i) \quad = \quad \{u : \max_{u \in U(i)} R(i,u) - R(i,u) \leq \delta \left| \max_{u \in U(i)} R(i,u) \right|, u \in U(i)\} \tag{83}$$

$$\min\ \lambda_R(r^*; \delta, w) \tag{84}$$

s.t.

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

$$\lambda_R(r^*;\delta,w) + \tilde{h}(i) = \begin{cases} \frac{w}{|\widetilde{U}(i)|} \sum_{u \in \widetilde{U}(i)} \left[g(i,u) + \sum_{j \in \mathcal{S}_{\mathcal{V}}} \tilde{p}_{ij}(u)\tilde{h}(j) \right] & \\ + \frac{1-w}{|U(i)|-|\widetilde{U}(i)|} \sum_{u \in U(i) \setminus \widetilde{U}(i)} \left[g(i,u) + \sum_{j \in \mathcal{S}_{\mathcal{V}}} \tilde{p}_{ij}(u)\tilde{h}(j) \right], & \\ & \text{if } |U(i)| \neq |\widetilde{U}(i)| \\ \frac{1}{|\widetilde{U}(i)|} \sum_{u \in \widetilde{U}(i)} \left[g(i,u) + \sum_{j \in \mathcal{S}_{\mathcal{V}}} \tilde{p}_{ij}(u)\tilde{h}(j) \right], & \text{if } |U(i)| = |\widetilde{U}(i)| \end{cases} \tag{85}$$

The parameter δ appearing in Equation 83 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 capacitated re-entrant line scheduling policy.

5.3 Selecting a Feature Space Φ for the Capacitated Re-entrant Line Scheduling Problem

5.3.1 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 the considered application context, the selected features are suggested by a number of queueing-theoretic concepts and results [6, 26] and they 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 [6, 26], similar information was employed for predicting performance bounds of queueing networks modeling re-entrant lines. Furthermore, the work of [46] 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 4. 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.[1] 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 capacitated re-entrant line.

Complexity of the suggested set Φ of feature functions The above feature specification results in 91 classes, including a total of $M(18M + 36L - 22) + L(18L - 35) + 7$

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

Table 4: Simple Features

Classes	Expressions
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	$I_{\{0\leq\sum_{j\in\sigma(W_i)}(n_{jw}+n_{jp}+n_{jb})\leq 0.2B_i\}}, i = 1, \ldots, L$
SF9	$I_{\{0.2B_i\leq\sum_{j\in\sigma(W_i)}(n_{jw}+n_{jp}+n_{jb})\leq 0.8B_i\}}, i = 1, \ldots, L$
SF10	$I_{\{0.8B_i\leq\sum_{j\in\sigma(W_i)}(n_{jw}+n_{jp}+n_{jb})\leq B_i\}}, i = 1, \ldots, L$
SF11	$I_{\{\sum_{j\in\sigma(W_i)}(n_{jw}+n_{jp}+n_{jb})=B_i\}}, i = 1, \ldots, L$
SF12	$B_i - \sum_{j\in\sigma(W_i)}(n_{jw}+n_{jp}+n_{jb}), i = 1, \ldots, L$

feature functions.[2] 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 5.4.

Using these feature functions, a linearly parameterized approximation function as defined in Equation 60 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 69 – 72. The evaluation of the approximating capability of feature space Φ was performed through the following numerical experiment.

5.3.2 A Numerical Experiment for Evaluating Φ

Design of a numerical experiment We tested the potential performance of the approximating architecture generated by the aforementioned feature functions, on two types

[2]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 5: Considered system configurations for a numerical experiment

Config.	number of workstations	number of job stages (JS) and job routes	buffering 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)

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 chapters, 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 5 summarizes the system configurations used in this experiment. For each configuration, 30 *problem instances* with randomly generated processing rates were considered. The number of states generated in each case, and the number of the employed feature functions, are summarized in Table 6. Notice that, since we are considering re-entrant lines consisting of single-server workstations, feature classes SF2 and SF5 are the same; we included only one of them, resulting in 78 classes, including a total of $\frac{M(25M+60L-35)}{2} + L(18L-35) + 7$ feature functions.

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 5.1. 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

$$\%error = \frac{Optimal\ TH - TH\ by\ rand.\ policy}{Optimal\ TH} \times 100. \tag{86}$$

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

Table 6: The number of states and feature functions in Φ for considered re-entrant lines

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

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 7 and 8 summarize the obtained results. More specifically, Table 7 lists the average, minimum and maximum % errors of throughput obtained by using the aforementioned heuristics on each of the six CRL configurations, while Table 8 reports the results characterizing the performance of the randomized policy obtained through the method of Section 5.2. Columns 2 and 3 in Table 8 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. Finally, Column 8 provides a measure of the "non-greediness" of the derived policy, by reporting the extra number of control actions included in $\widetilde{U}(i)$, averaged over all states $i \in \mathcal{S}_\mathcal{V}$.

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:

- 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. For the case of Configuration 1, the employed architecture supports perfect goodness-of-fit, but the reported throughput error is non-zero due to the randomizing nature of the derived

Table 7: Performance of heuristics for considered re-entrant lines

Config.	% error	FBFS	LBFS	FIFO	LWNQ-FBFS	LWNQ-LBFS	LWNQ-FIFO
Conf 1	Avg.	0	2.819378	0	0	0	0
	Min.	0	0.536441	0	0	0	0
	Max.	0	6.462280	0	0	0	0
Conf 2	Avg.	2.635282	3.056652	2.635282	2.060173	2.767874	2.060173
	Min.	0.002057	0.000411	0.002057	0.000206	0.000411	0.000206
	Max.	6.013643	11.416467	6.013643	7.432376	8.677738	7.432376
Conf 3	Avg.	1.022870	1.745342	1.022870	1.030130	1.161963	1.030130
	Min.	0	0	0	0	0	0
	Max.	4.301574	9.982387	4.301574	5.042340	5.198791	5.042340
Conf 4	Avg.	0.596465	3.315451	0.596465	0.596465	0.596465	0.596465
	Min.	0	0.097459	0	0	0	0
	Max.	2.831750	14.249711	2.831750	2.831750	2.831750	2.831750
Conf 5	Avg.	3.295272	3.330957	3.295272	1.987549	2.252614	1.987549
	Min.	0.000875	0.000875	0.000875	0.000175	0.000875	0.000175
	Max.	8.885160	14.164897	8.885160	6.956313	7.461679	6.956313
Conf 6	Avg.	2.921983	1.873558	2.921983	1.278979	1.421619	1.278979
	Min.	0.016242	0.000126	0.016242	0	0.000126	0
	Max.	11.611800	6.833881	11.611800	5.825995	6.906104	5.825995

Table 8: 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

policy.

- Even more importantly, this randomized policy is more consistent in its performance than the considered heuristics, as manifested by the reported maximum % errors.

- 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) can be found in Appendix A. More specifically, Tables 19 – 24 represent a range of values for δ and w that resulted in better performance than the considered heuristics. As it can be seen from these tables, in general, 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].

- The reported non-zero value for the averaged value of ϵ^* for the cases of Configurations 2, 4 and 5, when combined with the data of Table 6, 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.

5.3.3 Consideration of Scalability

The experiment reported in the previous section employed quite small system configurations, in an effort to maintain computational 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 LP formulations of Equations 51 – 54, Equations 69 – 72, and Equations 84 – 85, that are employed by 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 LP formulations of Equations 51 – 54, Equations 69 – 72, and Equations 84 – 85 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 the previous section, and the second generated by increasing the number of workstations and job stages.[3] 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

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

Table 9: Considered system configuration for the scalability test

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

Table 10: The number of states and feature functions in the considered system configuration for the scalability test

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

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 9 and 10 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 11. 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 (ii) 10 problem instances with randomly generated processing rates due to the long computational times involved.

A numerical experiment for Configurations 7 and 8 Tables 11 and 12 report respectively the results characterizing the performance of the randomized policy obtained through the method in Section 5.2 and the performance of some heuristics, when applied on Configurations 7 and 8. These results are consistent with the ones obtained in the previous section 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. From a more qualitative standpoint, they indicate that the considered set Φ of feature

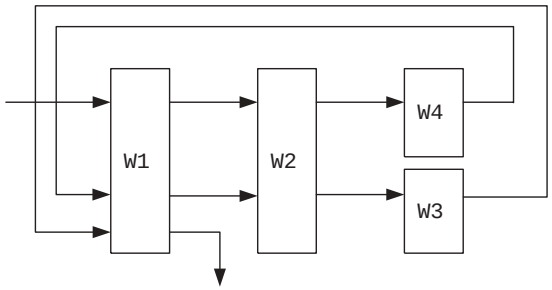


Figure 11: A re-entrant line with 4 single-server workstations and 7 job stages

Table 11: 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

functions has a good representational capability even for the case that the number of states is quite greater than the number of feature functions. Tables 25 and 26 in Appendix B.1 show that, similar to the case of smaller-sized re-entrant lines, the above results are robust with respect to the values of the parameter vector (δ, w). Finally, notice that the average values of the optimal approximation error, ϵ^*, are increased compared to those in the previous section. 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.

A numerical experiment for Configuration 9 Tables 13 and 14 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

Table 12: 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 13: 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

randomly selected processing rates and we fixed the value of parameter w to $w = 0.99$,[4] 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. The complete set of results characterizing the performance of the randomized policy on this configuration is provided in Table 27 of Appendix B.2.

[4]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 14: 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

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

5.4.1 Extending the Feature Space Φ by including up to 3-order Interactions

This section investigates the impact of adding 3-order interactions in Φ on the performance of the employed approximating scheme. A new extended set Φ' 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 Φ' 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 the previous section, 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.

5.4.2 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 the previous section for the evaluation of the feature set Φ. The number of the generated feature functions is summarized in Table 15. As in the previous section, 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 outlined in Section 5.1, while considering the same values of (δ, w) as in the previous section, and the obtained results are presented in Table 16. Some interesting remarks regarding these results can be summarized as follows:

Table 15: The number of states and feature functions in $\Phi^{'}$ for considered re-entrant lines

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

Table 16: Performance of the randomized policy generated by the proposed architecture using $\Phi^{'}$

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

- The performance achieved by using the new set $\Phi^{'}$ of feature functions was improved, as indicated by lower averaged value of ϵ^* and % errors, and maximum % errors, compared to those attained by the set Φ.

- 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, as indicated by the sensitivity analysis reported in Appendix C.

- The inclusion of higher-order interactions introduces new linearly independent feature functions, and results in small averaged values of ϵ^* in all configurations.

Figures 12 – 17 provide a graphical representation of the results tabulated in Tables 8 and 16. In these figures, the range of minimum and maximum % errors is represented as a straight line, and the average % error is marked with a dot on the line for each policy.

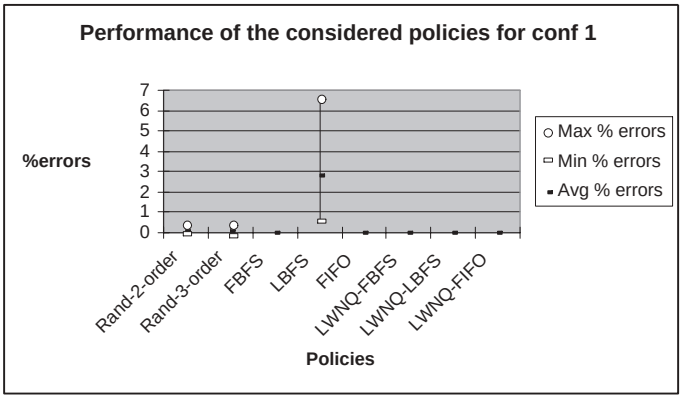


Figure 12: Performance of the considered policies for conf 1

Notice that, with the exception of Configuration 1[5], the randomized policy considering feature functions up to 3-order interactions has the lowest % errors in terms of the average and maximum % errors.

5.4.3 Statistical Significance of the Impact of Adding 3-order Interactions

The above claims about the performance improvement resulting by adding 3-order interactions in the employed feature set can be further formalized by testing the statistical significance of the mean difference of the average % errors resulting by considering sets Φ and $\Phi^{'}$. This test is performed as follows:

Since both numerical experiments reported in Sections 5.3 and 5.4 were performed with the same set of problem instances for all configurations, we have 180 paired observations of the % errors corresponding to the selected parameter set (δ, w) for each configuration, which gives best performance of the randomized policy. Table 17 summarizes the 180 paired % errors. By defining a single derived variable D as the difference between the paired values on % errors resulting from using Φ and $\Phi^{'}$, we can compute the observed sample mean

[5]In this case, the employed architecture supports perfect goodness-of-fit, but the reported throughput error is non-zero due to the randomizing effect of the derived policy.

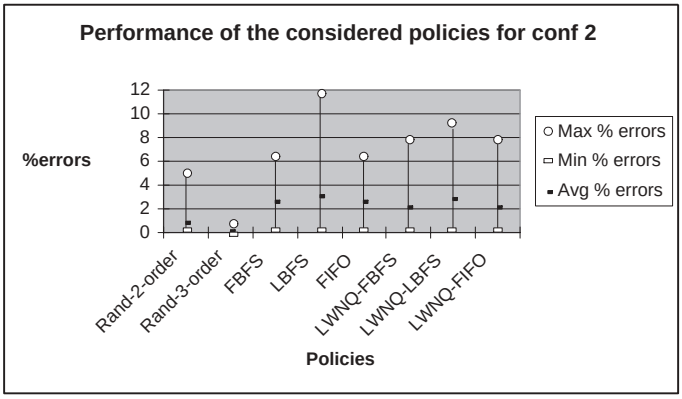


Figure 13: Performance of the considered policies for conf 2

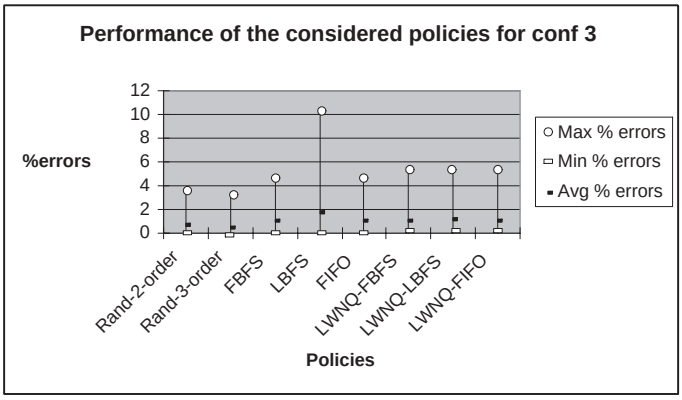


Figure 14: Performance of the considered policies for conf 3

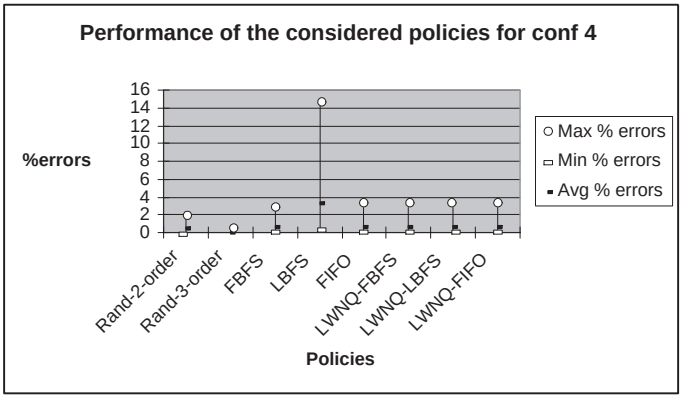


Figure 15: Performance of the considered policies for conf 4

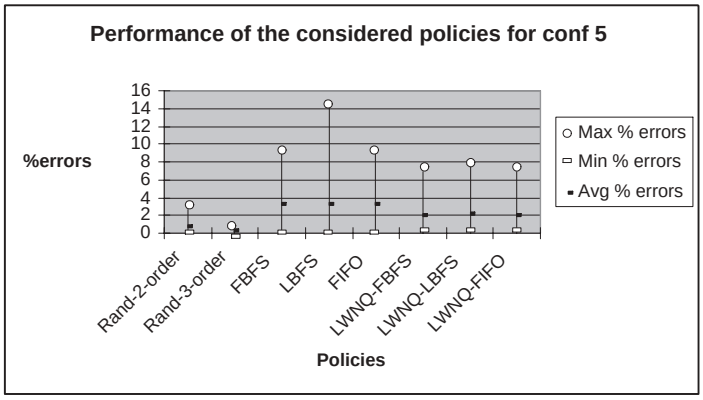


Figure 16: Performance of the considered policies for conf 5

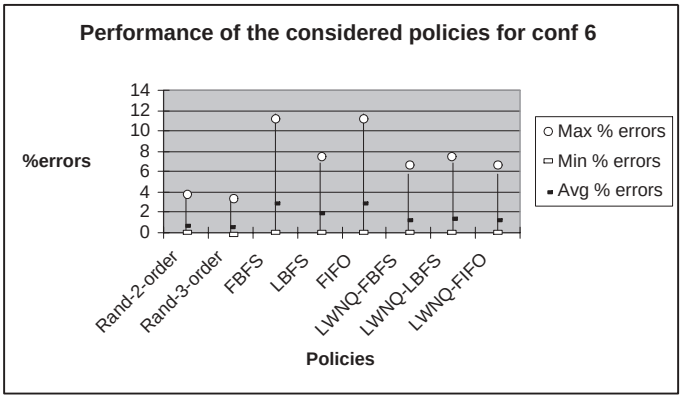


Figure 17: Performance of the considered policies for conf 6

difference $\bar{D}$ and sample standard deviation s_D as follows:

$$\bar{D} = \frac{1}{n}\sum_{i=1}^{n} D_i \tag{87}$$

$$S_D = \sqrt{\frac{\sum_{i=1}^{n}(D_i - \bar{D})^2}{n-1}} \tag{88}$$

where $n = 180$ and D_i is the difference of the % error of the i-th paired observation. We notice that, by the Central Limit Theorem [19], the sample mean difference $\bar{D}$ has a normal distribution with unknown variance. Then, we can perform the t-test [19] with 179 degrees of freedom by establishing the hypotheses set

$$H_0 : \mu_D \leq 0 \tag{89}$$

$$H_1 : \mu_D > 0 \tag{90}$$

in which the hypothesis $H_1 : \mu_D > 0$ corresponds to the tested assumption that the performance of the approximations provided by feature set Φ' is significantly better than the performance of the approximations provided by feature set Φ. The test statistic t is defined as

$$t = \frac{\sqrt{n}\bar{D}}{S_D}. \tag{91}$$

The sample mean difference is $\bar{D} = 0.374860$ and the sample standard deviation is $S_D = 0.749859$, and by Equation 91, $t = 6.706960 > t_{0.0005,179} = 3.346$. This result shows that

the hypothesis $H_1 : \mu_D > 0$ can be accepted with a confidence level higher than 99.95 % and manifests the significant improvement in the average % errors obtained by including 3-order interactions in Φ.

Table 17: The % errors of 180 paired observations corresponding to the selected set of parameters (δ, w) for each configuration

Config.	Run id	% errors by Φ	% errors by Φ'	Run id	% errors by Φ	% errors by Φ'	Run id	% errors by Φ	% errors by Φ'
	1	0.144743	0.144743	11	0.148691	0.148691	21	0.083249	0.083249
	2	0.147466	0.147466	12	0.031014	0.031014	22	0.166779	0.166779
	3	0.054883	0.054883	13	0.053890	0.053890	23	0.121420	0.121420
	4	0.117788	0.117788	14	0.059874	0.059874	24	0.065980	0.065980
Conf 1	5	0.120491	0.120491	15	0.024190	0.024190	25	0.031841	0.031841
	6	0.092400	0.092400	16	0.044980	0.044980	26	0.106633	0.106633
	7	0.160248	0.160248	17	0.089381	0.089381	27	0.089442	0.089442
	8	0.020716	0.020716	18	0.131275	0.131275	28	0.099126	0.099126
	9	0.143388	0.143388	19	0.077503	0.077503	29	0.117950	0.117950
	10	0.016866	0.016866	20	0.055555	0.055555	30	0.173766	0.173766
	1	0.057653	0.005134	11	0.175587	0.093135	21	2.652188	0.215701
	2	1.181942	0.049311	12	0.011035	0.003147	22	0.864900	0.089379
	3	4.886354	0.192549	13	0.103190	0.019424	23	0.146969	0.078152
	4	0.003095	0.044861	14	2.347488	0.086827	24	0.024128	0.009824
Conf 2	5	0.207850	0.126205	15	0.242229	0.034226	25	0.310429	0.130503
	6	0.127197	0.049554	16	0.093190	0.029894	26	0.337106	0.065899
	7	0.320962	0.037659	17	0.451198	0.069056	27	0.140199	0.081924
	8	0.200501	0.090242	18	2.914960	0.020081	28	0.998489	0.106058
	9	0.804243	0.007627	19	2.531976	0.115252	29	1.596371	0.060879
	10	0.006994	0.000828	20	0.506899	0.052212	30	0.357293	0.059529
	1	0.126364	0.094343	11	0.878933	1.405405	21	1.477685	0.510969
	2	1.442225	0.003745	12	0.089787	0.039812	22	0.482730	0.001138
	3	0.326428	0.178136	13	0.323682	1.005795	23	0.000895	0.001444
	4	0.108774	0.083155	14	1.311337	0.035553	24	0.342344	0.210012
Conf 3	5	0.086860	0.048910	15	0.314295	0.093443	25	0.057482	0.131598
	6	0.125900	0.048450	16	0.663470	0.351607	26	0.000104	0.000305
	7	1.515639	0.001495	17	2.771016	1.409424	27	0.272135	0.289310
	8	0.211804	0.066114	18	3.118610	3.184219	28	3.523133	1.837759
	9	0.819324	0.763269	19	0.042754	0.014221	29	0.129781	0.108519
	10	0.780008	0.669978	20	0.038846	0.012190	30	0.067636	0.000689
	1	0.079631	0.032985	11	0.330836	0.049467	21	1.908183	0.109073
	2	0.261469	0.031505	12	0.218634	0.045736	22	0.213555	0.053852
	3	0.382111	0.021966	13	0.207746	0.046439	23	0.216830	0.048303
	4	1.803840	0.193298	14	0.832170	0.039108	24	0.128076	0.022806
Conf 4	5	0.167630	0.016136	15	0.104039	0.001293	25	0.173007	0.034866
	6	0.407651	0.019034	16	0.259718	0.045447	26	0.169276	0.008482
	7	0.271557	0.096131	17	1.797923	0.042187	27	0.065672	0.065672
	8	0.379290	0.001023	18	0.255196	0.005842	28	0.339931	0.074400
	9	0.289841	0.012743	19	0.632535	0.032509	29	0.825471	0.058120
	10	1.398624	0.041048	20	1.573078	0.077435	30	0.065391	0.065391
	1	1.169046	0.158148	11	0.230686	0.393030	21	0.476956	0.207717
	2	0.081583	0.077790	12	0.389248	0.214816	22	0.174127	0.016210
	3	2.617387	0.353528	13	0.003318	0.005249	23	0.021502	0.016753
	4	0.373276	0.275595	14	0.316511	0.106810	24	0.621908	0.392162
Conf 5	5	0.879415	0.214636	15	0.266255	0.174166	25	0.815060	0.505576
	6	2.192346	0.289775	16	0.574802	0.408010	26	0.227492	0.258086
	7	0.194354	0.058829	17	0.250199	0.144090	27	0.178364	0.176580
	8	0.351379	0.205496	18	0.811196	0.260595	28	0.057391	0.042305
	9	2.089259	0.417429	19	1.926772	0.681049	29	0.995828	0.266162
	10	1.391036	0.110838	20	1.951421	0.692937	30	0.079929	0.071040
	1	0.253831	0.212555	11	1.059268	3.402330	21	1.876127	1.173183
	2	0.727746	0.299524	12	0.015971	0.020706	22	0.025017	0.010652
	3	0.808815	0.052520	13	2.416559	0.385687	23	0.544105	0.117746
	4	0.199053	0.064084	14	0.007535	0.002053	24	1.042744	0.694358
Conf 6	5	0.112465	0.059588	15	0.697257	0.546632	25	3.502581	2.276016
	6	0.021026	0.013442	16	0.282140	0.079646	26	2.627276	0.169124
	7	2.629283	0.592060	17	1.109867	0.666522	27	0.345717	0.311169
	8	0.011912	0.638690	18	0.000625	0.002393	28	0.436972	1.478034
	9	0.378821	0.280972	19	0.420825	0.280972	29	0.011889	0.009407
	10	0.194658	0.169844	20	0.033242	0.171497	30	0.036881	0.312283

CHAPTER 6

OPTIMAL DEADLOCK RESOLUTION IN RESOURCE ALLOCATION SYSTEMS

This chapter reports an additional set of results developed in this research program that, even though not directly related to the performance control problem addressed in this thesis, are connected to it from a methodological standpoint, since they capitalize upon the models and methods presented in Chapter 3. The key question addressed in this part of the work is as follows: While, as reported in Section 2.1, deadlock avoidance constitutes a key deadlock resolution strategy that can be adopted in the operation of contemporary RAS applications, it is not the only one. An alternative approach, known as *detection and recovery*, would let the problem of deadlock occur, and subsequently it would react to it through some exception handling procedure. In fact, one can conjecture a mixing approach where some deadlocks are systematically avoided while some others are detected and recovered from. In the following, first we provide a formal characterization of the alternative deadlock resolution strategies in the FSA context modeling the RAS behavior (c.f. Section 2.1) and subsequently we formulate and study the optimal deadlock resolution problem. The results presented in this chapter have appeared in [42].

6.1 Performance Modeling of Deadlock Resolution Strategies in Resource Allocation Systems

Modeling the alternative deadlock resolution strategies Using the formalism of Section 2.1, under the *detection and recovery* strategy, the system is allowed to access its entire reachable subspace S_r. Furthermore, whenever a deadlock is reached, the involved processes are identified, and the deadlock is resolved by *swapping* (a subset of) the deadlocked processes in a way that it will allow their further progress. In the FSA context, this swapping mechanism corresponds to a (single) transition from the deadlocked state s_d to

another deadlock-free state s'. Furthermore, since s' is reached through some exception handling procedure, it will be that $s' \in S \backslash S_r$. From s', the autonomous "normal" operation of the system is resumed, until the system reaches another deadlocked state, in which case, the deadlock detection and recovery scheme described above is repeated on this new state. Hence, in the FSA modeling framework, the deadlock detection and recovery approach establishes the ability of the system to run to completion, even under the occurrence of deadlocks, through the insertion of additional transitions to the STD modeling the original feasible system behavior, that correspond to the deadlocked process swaps by some sort of exception handling routine. In other words, the insertion of these new transitions ensures that for every RAS state $s \in S$, visited by the system when operated under the deadlock detection and recovery strategy, it holds: $s \xrightarrow{*} s_0$.

The above operational scheme can be extended to the *randomized* deadlock avoidance strategy, which operates similar to the detection and recovery approach, with the additional feature that resource allocation requests corresponding to transitions $t: \ s \rightarrow s', \ \text{with} \ s \in S_{rs} \ \wedge \ s' \in S_{ru}$, are satisfied only with a certain probability ω_t. In particular, assuming that $\omega_t \neq 0, \ \forall \ t$, the reachable state space for a given RAS configuration under the randomized deadlock avoidance strategy is identical to the corresponding state space that is reachable when the system is operated under the detection and recovery approach. Hence, randomized deadlock avoidance establishes a continuum between the two extreme strategies of detection and recovery and "classical" deadlock avoidance.

RAS performance modeling and optimization Under the assumption that the timing of the various events identified in the STD modeling the system behavior under a given deadlock resolution strategy is *exponentially* distributed, the system *timed* dynamics can be effectively modeled by a *Continuous Time Markov Chain (CTMC)* [7]. To formalize the subsequent development, consider a given RAS configuration, controlled by a *randomized DAP (R-DAP)* $\mathcal{P}$. Let $S_r(\mathcal{P}) \subseteq S$ denote the system reachable subspace under the considered R-DAP. Furthermore, for every transition $t_{ij}: \ s_i \xrightarrow{\mathcal{P}} s_j, \ s_i, s_j \in S_r(\mathcal{P})$, that is feasible

under the considered policy, let $\bar{q}_{ij}$ denote the rate of the exponential distribution characterizing the *natural* timing of the corresponding event. In the considered operational context of flexibly automated manufacturing systems, $\bar{q}_{ij}$ correspond to the job arrival/loading, processing and unloading rates, as well as the rates characterizing the job swapping mechanism during the deadlock recovery phase, and they are determined by the system processes and its operational environment. However, under the control of R-DAP $\mathcal{P}$, the occurrence rate of transitions $t_{ij}: \ s_i \rightarrow s_j, \ \text{with } s_i \in S_{rs} \ \wedge \ s_j \in S_{ru}$, is moderated by the transition control probability $\omega_{ij} \in [0,1]$ to $q_{ij} = \omega_{ij} \cdot \bar{q}_{ij}$. Hence, the eventual *infinitesimal generator matrix* Q, defining the CTMC that describes the system dynamics when it is controlled by R-DAP $\mathcal{P}$, is given by $Q = [q_{ij}]$ with

$$q_{ij} = \begin{cases} \omega_{ij}\bar{q}_{ij} & \text{if } s_i, \ s_j \in S_r(\mathcal{P}) \ \wedge i \neq j \ \wedge \ s_i \xrightarrow{\mathcal{P}} s_j \ \wedge s_i \in S_{rs} \ \wedge \ s_j \in S_{ru} \\ \bar{q}_{ij} & \text{if } s_i, \ s_j \in S_r(\mathcal{P}) \ \wedge \ i \neq j \ \wedge \ s_i \xrightarrow{\mathcal{P}} s_j \ \wedge (s_i \notin S_{rs} \ \vee \ s_j \notin S_{ru}) \\ 0 & \text{if } \ s_j \in S_r(\mathcal{P}) \ \wedge i \neq j \ \wedge \ s_i \not\xrightarrow{\mathcal{P}} s_j \\ -\sum_{j:\ j \neq i} q_{ij} & \text{if } s_i \in S_r(\mathcal{P}) \ \wedge \ i = j. \end{cases} \tag{92}$$

In the formalism of Equation 92, the system dynamics under the control of (classical) deadlock avoidance (resp., detection and recovery) strategy, are modeled by setting $\omega_{ij} = 0$ (resp., 1) $\forall \ (i,j): \ s_i \rightarrow s_j$ with $s_i \in S_{rs} \ \wedge \ s_j \in S_{ru}$. Furthermore, since, under any deadlock resolution strategy, the resulting system behavior is irreducible, aperiodic, finite-state, and therefore, ergodic, the CTMC defined by Equation 92 has a *unique limiting stationary distribution*, expressed by the *steady state probability vector* π, obtained by the following system of equations [7]:

$$\pi^T Q = \mathbf{0}^T, \tag{93}$$

$$\sum_{\{i:\ s_i \in S_r(\mathcal{P})\}} \pi_i = 1. \tag{94}$$

Given the stability implied by the ergodic nature of the system behavior, a characterization of the steady-state (long-run) system throughput can be obtained by considering the cumulative rate according to which jobs are loaded into the system. Therefore, recognizing that the steady state probabilities π_i can be interpreted as the percentage of time that the

RAS spends in each state $s_i \in S_r(\mathcal{P})$, while element q_{ij} denotes the rate according to which the system transitions from state s_i to state s_j, once in state s_i, the cumulative job loading rate expressing the system throughput, under R-DAP $\mathcal{P}$, is given by:

$$TH(\mathcal{P}) = \sum_{\{(i,j):\ \text{transition } s_i \xrightarrow{\mathcal{P}} s_j \text{ corresponds to a job loading event}\}} \pi_i q_{ij}. \quad (95)$$

Finally, Equations 92 – 95, combined with the fact that, in the considered modeling framework, a R-DAP($\mathcal{P}$) is essentially defined by the values assigned to the probabilities controlling the transition rates from the safe to the unsafe region of the underlying RAS – to be collectively denoted by the vector ω – imply that the *optimal deadlock resolution strategy selection problem* can be formally stated as follows:

$$\max_{\omega}\ TH(\omega; \bar{q}_{ij}) = \sum_{\{(i,j):\ \text{transition } s_i \xrightarrow{\mathcal{P}} s_j \text{ corresponds to a job loading event}\}} \pi_i q_{ij} \quad (96)$$

s.t.

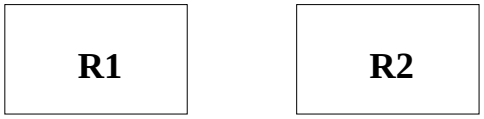
$$\pi^T Q(\omega; \bar{q}_{ij}) = \mathbf{0}^T \quad (97)$$

$$\sum_{i:\ s_i \in S_r(\mathcal{P})} \pi_i = 1 \quad (98)$$

$$\forall\ i, j,\ \omega_{ij} \in [0, 1]. \quad (99)$$

Example To provide a concrete example of the concepts introduced above, consider the small RAS depicted in Figure 18. This RAS consists of two resources, R_1 and R_2 of unit capacity, and it supports the execution of two job types, JT_1 and JT_2, with respective process plans $JT_1 : < [1,0]^T, [0,1]^T >$, $JT_2 : < [0,1]^T, [1,0]^T >$.[1] The STD modeling the system state space under the control of the three deadlock resolution strategies considered in this work is depicted in Figure 19. Specifically, the uncontrolled system behavior is modeled by the subgraph induced by the state subset $S_r = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$. Furthermore, the reachable safe subspace, admitted by the optimal deadlock avoidance policy is defined by the state subset $S_{rs} = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$, while the reachable unsafe subspace consists

[1] This RAS could model, for instance, the allocation of the buffering capacity in a two-machine robotic cell, or a two-chamber cluster tool, supporting the processing of two parts with counterflow process plans.



JT1: R1 -> R2

JT2: R2 -> R1

Figure 18: Example: The considered RAS

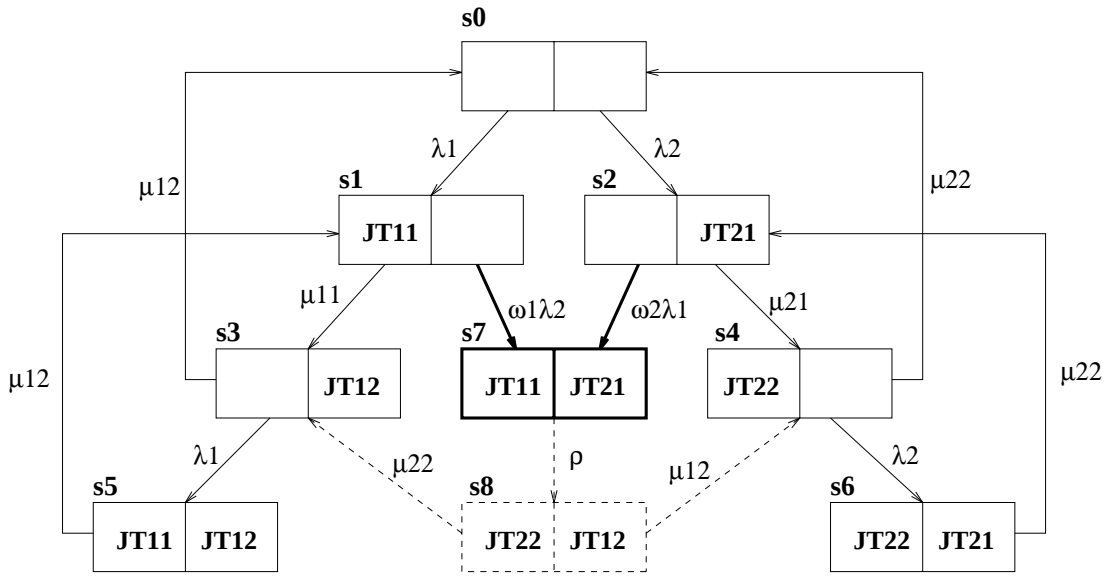


Figure 19: Example: The state space describing the system behavior under various deadlock resolution strategies

of the singleton $S_{ru} = \{s_7\}$, which constitutes the unique deadlock state. Hence, the transitions from S_{rs} to S_{ru}, that must be controlled under randomized deadlock avoidance, are $t_1^d = s_1 \rightarrow s_7$, and $t_2^d = s_2 \rightarrow s_7$. In the following, we shall denote the respective control probabilities by ω_1 and ω_2. Moreover, under the randomized deadlock avoidance and the detection and recovery strategies, the deadlock of state s_7 is resolved by swapping the two deadlocked jobs, and the resulting state, s_8, is a state that is unreachable under the "normal" system operation. Finally, once in state s_8, the system will return to its safe region S_{rs}, either through state s_3 or through state s_4.

In order to characterize the timed dynamics of this system and its expected throughput at steady state, let us further assume that the loading time for jobs of type $JT_i,\ i = 1, 2$, follows an exponential distribution with rate $\lambda_i,\ i = 1, 2$. Similarly, the processing time for stage $JT_{ij},\ i, j = 1, 2$, is exponentially distributed with rate μ_{ij}. Finally, the time required for recovering from the deadlock of state s_7 is exponentially distributed with rate ρ. Then, the occurrence rates for the various system transitions are annotated in Figure 19, while the infinitesimal matrix generator for the resulting CTMC is as follows:

$$Q = \begin{bmatrix}
-(\lambda_1+\lambda_2) & \lambda_1 & \lambda_2 & & & & & & \\
 & -(\mu_{11}+\omega_1\lambda_2) & & \mu_{11} & & & \omega_1\lambda_2 & & \\
 & & -(\mu_{21}+\omega_2\lambda_1) & & \mu_{21} & & \omega_2\lambda_1 & & \\
\mu_{12} & & & -(\mu_{12}+\lambda_1) & & \lambda_1 & & & \\
\mu_{22} & & & & -(\mu_{22}+\lambda_2) & & \lambda_2 & & \\
 & \mu_{12} & & & & -\mu_{12} & & & \\
 & & \mu_{22} & & & & -\mu_{22} & & \\
 & & & & & & & -\rho & \rho \\
 & & & \mu_{22} & \mu_{12} & & & & -(\mu_{22}+\mu_{12})
\end{bmatrix}. \tag{100}$$

Denoting the system steady state probabilities by $\pi_i,\ i = 0, \ldots, 8$, the cumulative job loading rate at steady state, defining its (long-run) throughput, is given by:

$$TH = \lambda_1\pi_0 + \lambda_2\pi_0 + \omega_1\lambda_2\pi_1 + \omega_2\lambda_1\pi_2 + \lambda_1\pi_3 + \lambda_2\pi_4. \tag{101}$$

Therefore, the problem of selecting the optimal deadlock resolution strategy for this example

system is posed as follows:

$$\max_{\omega_1,\omega_2} TH(\omega_1,\omega_2;\lambda_i,\mu_{ij},\rho,\ i,j=1,2) = \lambda_1\pi_0+\lambda_2\pi_0+\omega_1\lambda_2\pi_1+\omega_2\lambda_1\pi_2+\lambda_1\pi_3+\lambda_2\pi_4 \quad (102)$$

s.t.

$$\pi^T Q(\omega_1,\omega_2;\lambda_i,\mu_{ij},\rho,\ i,j=1,2) = \mathbf{0}^T \quad (103)$$

$$\sum_{i=0}^{8}\pi_i = 1.0 \quad (104)$$

$$\omega_1,\ \omega_2 \in [0,1]. \quad (105)$$

$\diamondsuit$

6.2 The Structure of the Optimal Deadlock Resolution Strategy

In this section, it is shown that the problem of selecting the optimal deadlock resolution strategy, when formulated according to Equations 96 – 99, has always an optimal solution that is an *extreme point* – i.e., a corner point – of the hypercube $[0,1]^{\dim(\omega)}$. This result is developed in two main steps: (i) First, the functional dependence of the objective function of Equation 96 on the problem decision variables ω_{ij}, under constraints (97) and (98), is analytically characterized. (ii) Subsequently, it is established that the derived functional form, when constrained in the hypercube $[0,1]^{\dim(\omega)}$, always possesses a maximal value that is an extreme point of the considered domain.

The result relating to the first of the aforementioned steps is formally stated and proven as follows:

Proposition 1 *The optimization problem of Equations 96 – 99 can be transformed to an equivalent optimization problem of the form:*

$$\max_{\omega} TH(\omega;\bar{q}_{ij}) = \frac{N(\omega;\bar{q}_{ij})}{D(\omega;\bar{q}_{ij})} \quad (106)$$

s.t.

$$\forall\ i,j,\ \omega_{ij} \in [0,1], \quad (107)$$

where $N(\omega;\bar{q}_{ij})$ and $D(\omega;\bar{q}_{ij})$, appearing in Equation 106, are first-degree polynomials with respect to each of the decision variables ω_{ij}.

Proof: Since the CTMC defined by matrix Q is ergodic, it possesses a unique stationary distribution, π, which can always be effectively computed by solving the following system of equations [7]:

$$\pi^T \left[\begin{array}{cc} \hat{Q}(\omega;\bar{q}_{ij}) & \mathbf{1} \end{array} \right] = \left[\begin{array}{cc} \mathbf{0^T} & 1 \end{array} \right]. \tag{108}$$

In Equation 108, $\hat{Q}(\omega;\bar{q}_{ij})$ denotes the matrix obtained from the chain infinitesimal generator $Q(\omega;\bar{q}_{ij})$ by removing its first column, corresponding to state s_0.[2] To facilitate the subsequent discussion, let us rewrite Equation 108 in the most familiar form of:

$$\left[\begin{array}{c} \hat{Q}^T(\omega;\bar{q}_{ij}) \\ \mathbf{1}^T \end{array} \right] \pi = \left[\begin{array}{c} \mathbf{0} \\ 1 \end{array} \right]. \tag{109}$$

Furthermore, let us denote by A the system matrix in the left-hand-side (lhs) of Equation 109. Then, regarding A, the following remarks hold: (i) A is a square invertible matrix, and therefore, $\det(A)$ will exist and it will always have a non-zero value. (ii) Since, by the problem definition, each control variable ω_{ij} is associated uniquely with an unsafe transition linking the safe to the unsafe region of the system state space, it appears in a unique row of $\hat{Q}$, and therefore, in a unique column of A. More specifically, ω_{ij} will appear in the column corresponding to the π_i component of the steady state probability vector.

The first of the above observations implies that the system steady state probabilities, π_k, can always be computed as functions of ω_{ij} and $\bar{q}_{ij}$, by means of Cramer's rule [49]:

$$\pi_k(\omega;\bar{q}_{ij}) = \frac{\det(A_k(\omega;\bar{q}_{ij}))}{\det(A(\omega;\bar{q}_{ij}))}, \tag{110}$$

where $A_k(\omega;\bar{q}_{ij})$ is the matrix obtained from $A(\omega;\bar{q}_{ij})$, by substituting its column corresponding to variable π_k with the right-hand-side (rhs) vector of Equation 109. The second of the above remarks regarding $A(\omega;\bar{q}_{ij})$, implies the following: (i) Both, the numerator, $\det(A_k(\omega;\bar{q}_{ij}))$ and the denominator, $\det(A(\omega;\bar{q}_{ij}))$, in the rhs of Equation 110, are first-degree polynomials with respect to each control variable ω_{ij}. (ii) The numerator, $\det(A_k(\omega;\bar{q}_{ij}))$, is independent of the particular ω_{ij} with $i = k$ (assuming that such an ω-variable exists in the original problem definition).

[2]Notice that, for a well-defined RAS, all transitions emanating from and/or leading to state s_0 are safe. Hence, under R-DAP control, $q_{ij} = \bar{q}_{ij}$ (c.f., Equation 92), and therefore, by dropping this first column of Q, we still maintain all the problem control variables, ω_{ij}.

The last two remarks further imply that each steady state probability π_k is functionally dependent on ω according to the fractional form specified by Proposition 1. Moreover, the second of these remarks implies that this functional form applies also to the products $\pi_k q_{kj}$, since it guarantees that in the case of controlled transitions t_{kj} corresponding to loading events, where $q_{kj} = \omega_{kj}\lambda_{kj}$, the numerator of π_k, $\det(A_k(\omega; \bar{q}_{ij}))$, is itself independent of ω_{kj}. Finally, since all products $\pi_k q_{kj}$ have the same denominator $\det(A(\omega; \bar{q}_{ij}))$, the functional form defined in Proposition 1 applies also to their summation. But then, the result of Proposition 1 is established simply by noticing that the problem objective function, defined in Equation 96, is just the summation of a pertinently selected subset of the product terms $\pi_k q_{kj}$, while Equation 107 is simply the last constraint in the original problem formulation. $\diamondsuit$

The next lemma will be used as a stepping stone in order to prove that the optimization problem defined by Equations 106 – 107 has an optimal solution that is an extreme point of its feasible region.

Lemma 12 *The* single-variable *optimization problem:*

$$\max_x \; f(x) = \frac{ax + b}{cx + d} \tag{111}$$

s.t.

$$x \in [0, 1] \tag{112}$$

with the additional assumption that

$$cx + d \neq 0, \quad \forall \; x \in [0, 1] \tag{113}$$

always has an optimal solution in the set $\{0, 1\}$.

Proof: Since function $f(x)$ is well-defined over the interval $[0, 1]$, it can be differentiated, giving:

$$\frac{df(x)}{dx} = \frac{ad - bc}{(cx + d)^2}. \tag{114}$$

Then, assuming that $ad - bc \neq 0$, $\text{sign}(df(x)/dx) = \text{sign}(ad - bc), \; \forall \; x$, and therefore, $f(x)$ is a monotonically increasing or decreasing function, depending on whether $ad - bc$ has a

positive or negative value. In the first case, the optimization problem of Lemma 12 has the optimal solution of 1, while in the second case, its optimal solution is 0. Finally, in the remaining case that $ad - bc = 0$, $f(x)$ has a constant value over the interval $[0, 1]$, and therefore, both ends of this interval correspond to optimal solutions. $\diamondsuit$

Finally, the main result of this section is stated and proven in the following theorem:

Theorem 2 *The optimal deadlock resolution strategy selection problem, defined in Equations 96 – 99, always has an optimal solution that is an extreme point of the hypercube* $[0, 1]^{dim(\omega)}$.

Proof: We prove this result by contradiction, utilizing the transformed problem version of Proposition 1. Hence, suppose that *all* optimal solutions to the optimization problem defined by Equations 106 – 107 are interior points of the hypercube $[0, 1]^{\dim(\omega)}$.[3] Let ω^* denote such an optimal interior point. Then, there must exist a component ω^*_{kl} of ω^* such that $\omega^*_{kl} \in (0, 1)$. Consider the function $TH(\omega_{kl};\ \omega^*_{ij},\ i \neq k \vee j \neq l,\ \bar{q}_{ij})$. This function is a single-variable function possessing the fractional form defined in Lemma 12, and it is defined over the interval $[0, 1]$. Yet, under the considered hypothesis, its maximal value is obtained at $\omega^*_{kl} \in (0, 1)$, which contradicts the result of Lemma 12 (and establishes the truth of Theorem 2). $\diamondsuit$

Example Let us consider the example RAS introduced in Section 6.1, when the loading and processing rates take the following values: $\lambda_1 = 1.0$, $\lambda_2 = 1.0$, $\mu_{11} = 3.0$, $\mu_{12} = 2.0$, $\mu_{21} = 1.0$ and $\mu_{22} = 2.0$. Table 18 provides the analytical form of the system throughput function, $TH(\omega_1, \omega_2;\ \rho)$, as characterized by Equation 106, for three different values of the deadlock recovery rate, ρ. Figure 20 also provides the plots of these functions over their domain area $[0, 1]^2$. As expected, in all cases, the resulting throughput function obtains its maximum (and minimum) value at one of the extreme points of its domain; the detailed characterization of the optimal solution and the maximal value of the objective function are included in Table 18. It is interesting to notice that for the case of $\rho = 0.5$, the maximal

[3]Notice that (i) the functional form of $TH(\omega; \bar{q}_{ij})$, implied by Proposition 1, (ii) the fact that $D(\omega; \bar{q}_{ij}) \equiv \det(A(\omega; \bar{q}_{ij})) \neq 0,\ \forall\ \omega$, and (iii) the finiteness of the problem feasible region (c.f., Equation 107), imply that the considered problem is well-defined.

Table 18: Example: The system throughput as a function of the control variables, ω_1 and ω_2, and its maximal value, for various values of the deadlock recovery rate, ρ.

ρ	$TH(\omega_1, \omega_2)$	(ω_1^*, ω_2^*)	$TH(\omega_1^*, \omega_2^*)$
0.1	$\frac{6(12+7\omega_1+21\omega_2+12\omega_1\omega_2)}{108+166\omega_1+462\omega_2+399\omega_1\omega_2}$	$(0,0)$	0.667
0.5	$\frac{6(12+7\omega_1+21\omega_2+12\omega_1\omega_2)}{108+70\omega_1+174\omega_2+111\omega_1\omega_2}$	$(0,1)$	0.702
1.0	$\frac{6(12+7\omega_1+21\omega_2+12\omega_1\omega_2)}{108+58\omega_1+138\omega_2+75\omega_1\omega_2}$	$(1,1)$	0.823

throughput is obtained for $(\omega_1^*, \omega_2^*) = (0, 1)$, which is a strategy conceptually different from the two classical approaches of deadlock avoidance, and deadlock detection and recovery. However, as the deadlock recovery rate ρ increases (resp., decreases), the optimal strategy switches to deadlock detection and recovery (resp., deadlock avoidance), since the (time) cost of deadlock recovery becomes lower (resp., prohibitively higher) than the productivity gains obtained from the enhanced operational concurrency. For a more analytical treatment of the eventual policy convergence to deadlock avoidance (resp., detection and recovery), as the deadlock recovery rate(s) ρ decreases (resp., increases) in value, the reader is referred to [40]. $\diamondsuit$

Generalizing Theorem 2 Concluding this section, we notice that even though the result of Theorem 2 was developed for the case that the optimized objective was the (long-run) system throughput, it can be easily generalized to all variations of the formulation of Equations 96 – 99, where the optimized function is a linear combination of the system steady-state probabilities, π, provided that the coefficient multiplying the steady-state probability π_k is only a function of ω_{ij} with $i = k$. As a more concrete example, consider the objective of minimizing the average number of parts in the system, $\bar{N}$. It can be easily seen that, under steady-state operation, this statistic is given by:

$$\bar{N} = \sum_{i=0}^{S_r(\mathcal{P})} N(s_i)\pi_i, \tag{115}$$

where $N(s_i)$ denotes the number of parts in the system when it is in state $s_i \in S_r(\mathcal{P})$. Since $N(s_i)$ is independent of ω, it follows that Equation 115 is minimized, under the constraints of Equations 97 – 99, at one of the extreme points of the hypercube $[0, 1]^{\dim(\omega)}$. Furthermore,

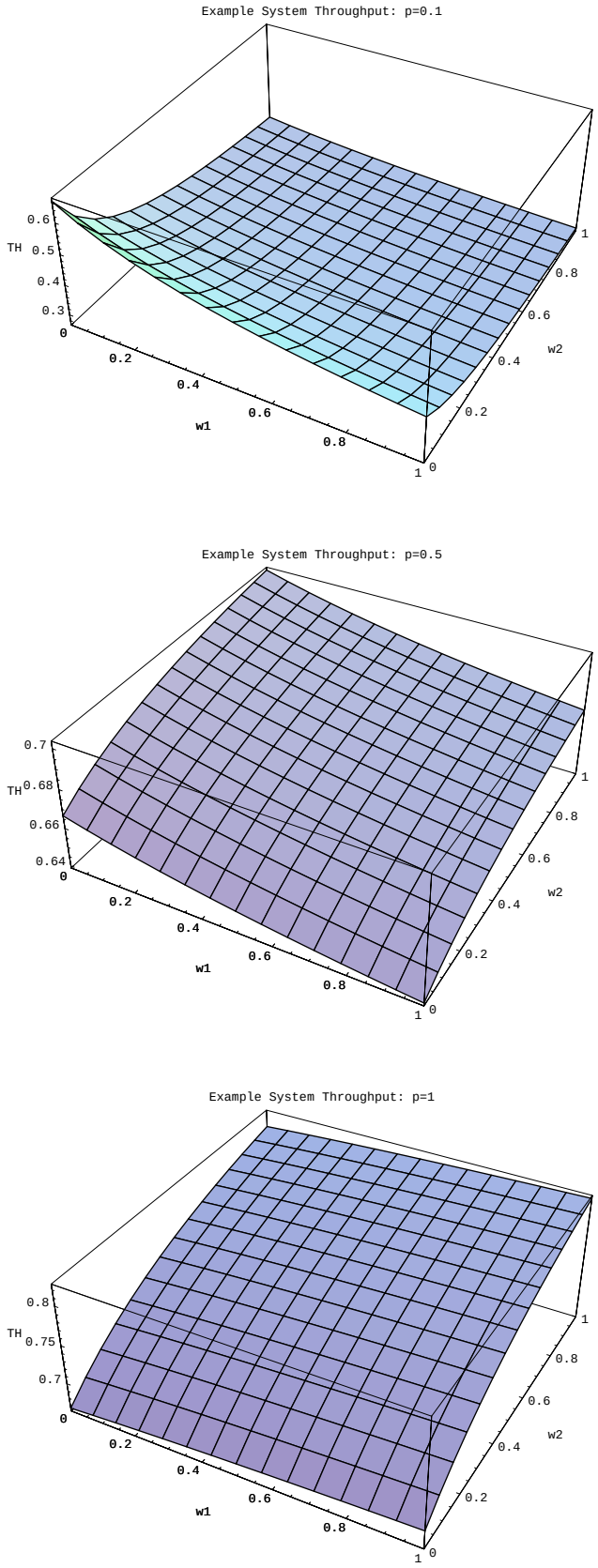


Figure 20: Example: Graphing the throughput function $TH(\omega_1, \omega_2;\ \rho)$ for the considered ρ values

noticing that, (i) by Little's law [7], the average job sojourn time, $\bar{D}$, can be expressed by

$$\bar{D} = \frac{\bar{N}}{TH} \tag{116}$$

and (ii) the fractional functional forms expressing the dependency of the quantities $\bar{N}$ and TH on the control variable vector ω (c.f., Equation 106) have the same denominator, $\det(A(\omega; \bar{q}_{ij}))$, it can be concluded that the functional dependence of the average sojourn time, $\bar{D}$, on the control vector, ω, is also expressed by a fractional form where, both, the numerator and the denominator are first-degree polynomials with respect to each control variable ω_{ij}. Hence, this important system performance measure, under the constraints of Equations 97 – 99, is also optimized at an extreme point of the hypercube $[0,1]^{\dim(\omega)}$.

6.3 Product-mix Considerations and Optimal Randomized-Deadlock Avoidance Policies

The formal arguments developed in the previous section in order to establish the non-essential role of randomization in the optimization of the applied deadlock resolution strategy with respect to the performance criteria of throughput, WIP inventories, and the job expected sojourn times, reveal also the key problem elements that underly this result. Specifically, Proposition 1 and Theorem 2 imply that the aforementioned result is based on (i) the ability to express the considered objective functions as a fraction of two multi-linear functions of the control variables, and (ii) the structure of the solution space, which is the entire hypercube $[0,1]^{\dim(\omega)}$. Furthermore, both of these problem properties result from the fact that the various control probabilities, ω_{ij} are mutually independent. It follows then that the negation of this mutual independence can lead to problem variations for which the randomization of the applied deadlock resolution strategy can be essential for achieving optimum performance.

As a case in point, in this section we consider the problem variation where the formulation of Equations 96 – 99 is augmented with an additional constraint of the type:

$$\frac{TH_k}{TH_l} = \xi_{kl}, \quad (k,l) \in \mathcal{C} \subseteq \mathcal{J} \times \mathcal{J}, \tag{117}$$

This constraint enforces a *"product-mix"* specification on the operation of the underlying

production system, and it is a requirement that arises naturally in many multi-item production systems either due to higher-level production planning taking place in the company, or due to the fact that the considered parts constitute components for a higher-level (sub-)assembly, produced in some downstream operation of the overall supply chain [35]. In the following, first we demonstrate, through an example, how the aforementioned type of constraint establishes some coupling among the problem control variables which restrains the feasible region for the original formulation of Equations 96 – 99, and leads to a randomized optimal solution for the underlying optimization problem. Furthermore, in the second part of the section, we establish an interesting topological property for the optimal solution(s) of the extended problem formulation addressed in this section.

Example To provide a concrete example of the way that Constraint 117 affects the feasible region of the original problem formulation of Equations 96 – 99, consider the example system of Figure 18, under the parameterization introduced in the example of Section 6.2, for the particular case that the deadlock recovery rate $\rho = 1.0$. We remind the reader that, for this particular system, the system total throughput depends on the control variables ω_1 and ω_2 as follows:

$$TH(\omega_1, \omega_2) = \frac{6(12 + 7\omega_1 + 21\omega_2 + 12\omega_1\omega_2)}{108 + 58\omega_1 + 138\omega_2 + 75\omega_1\omega_2}. \tag{118}$$

Furthermore, the partial throughput functions for each of the two job types can be obtained in a similar fashion, and they are as follows:

$$TH_1(\omega_1, \omega_2) = \frac{6(6 + 4\omega_1 + 9\omega_2 + 6\omega_1\omega_2)}{108 + 58\omega_1 + 138\omega_2 + 75\omega_1\omega_2}, \tag{119}$$

$$TH_2(\omega_1, \omega_2) = \frac{18(2 + \omega_1 + 4\omega_2 + 2\omega_1\omega_2)}{108 + 58\omega_1 + 138\omega_2 + 75\omega_1\omega_2}. \tag{120}$$

Hence, in this particular case, the product-mix constraint of Equation 117 takes the form:

$$\frac{6 + 4\omega_1 + 9\omega_2 + 6\omega_1\omega_2}{3(2 + \omega_1 + 4\omega_2 + 2\omega_1\omega_2)} = \xi. \tag{121}$$

For a value of $\xi = 0.9$, Equation 121 can be solved for ω_2, giving:

$$\omega_2 = -\frac{0.6 + 1.3\omega_1}{-1.8 + 0.6\omega_1}. \tag{122}$$

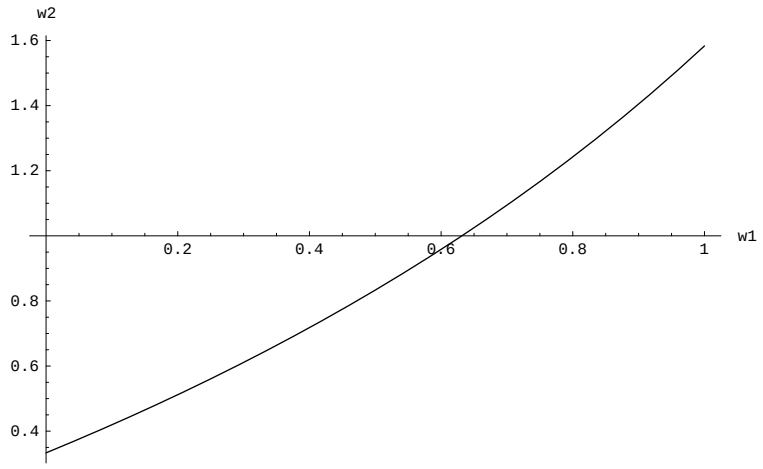


Figure 21: Example: Graphing the problem feasible region under the additional constraint of Equation 121, and with $\xi = 0.9$

.

A plot of Equation 122 for $\omega_1 \in [0, 1]$ is provided in Figure 21. Notice that Constraint 99 in the original problem formulation implies that the actual feasible region for the modified problem is the segment of the depicted curve contained among the points $(0, 1/3)$ and $(12/19, 1)$. Hence, plotting the function of Equation 118 for $\omega_1 \in [0, 12/19]$, and with ω_2 computed according to Equation 122, we obtain the curve depicted in Figure 22. From this figure, it can be deduced that, for the considered product-mix requirement, the system throughput is maximized by applying a randomized DAP with $(\omega_1, \omega_2) = (12/19, 1.0)$. $\diamondsuit$

Some interesting remarks regarding the presented example are as follows:

1. Notice that in the resulting optimal solution, $\omega_1 \in (0, 1)$. In essence, the randomization of the deadlock resolution strategy, by setting this control variable to a non-extreme value, is the mechanism used for accommodating the requirement of Equation 121 in the system operations. From a more mathematical standpoint, the constraint of Equation 121, introduces some coupling between the two initially independent control variables ω_1 and ω_2, which eventually reduces the degrees of freedom

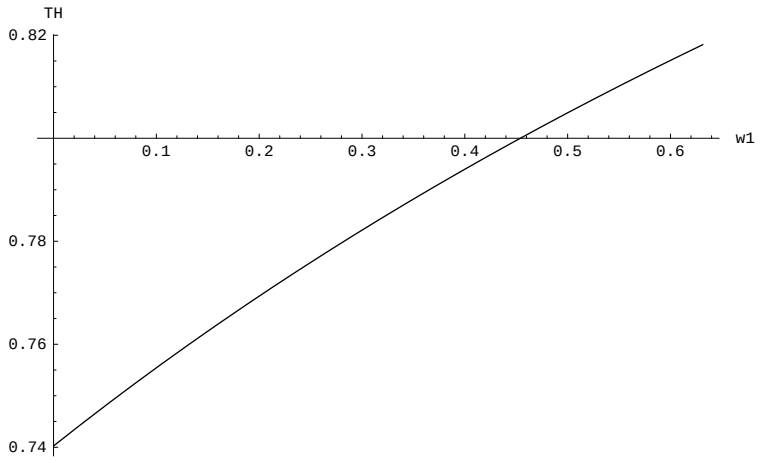


Figure 22: Example: Graphing the system (total) throughput as a function of the independent variable ω_1 and for a product-mix requirement $TH_1/TH_2 = 0.9$

.

of the original feasible region from 2 to 1. In other words, the feasible region for the modified problem formulation, $\mathcal{F}'$, reduces to a single curve in the original feasible region, $\mathcal{F} \equiv [0,1]^{dim(\omega)}$, which subsequently leads to the optimization of the considered formulation in an interior point of $\mathcal{F}$.

2. From a more practical standpoint, the resulting randomization of the deadlock resolution strategy is the mechanism used by the optimal policy in order to establish the required product-mix – i.e., the requirement of Equation 121 – in the system operations. In fact, it is possible that the problem formulation resulting by the addition of Constraint 117 to the original formulation of Equations 96 – 99, is infeasible. For an example, the reader can consider Equation 121 for $\xi = 2$. In this case, the accommodation of the product-mix constraint will require the control of additional transitions in the system STD, according to the randomized scheme presented above. In fact, the reader should convince herself that, in the most general case, the association of a control probability with every single transition of the underlying STD essentially

provides a *scheduling* mechanism for the considered RAS.

3. Finally, it should be noticed that the optimal policy for the above example was obtained at one of the two points constituting the *boundary* $\mathcal{B}$ of the feasible region $\mathcal{F}'$. In the considered problem context,[4] $\mathcal{B}$ is defined by the fact that some control variable(s) ω_{ij} takes an extreme value of its feasible interval $[0, 1]$, and further advancement beyond $\mathcal{B}$, in a way that satisfies Constraint 117, will eventually violate Constraint 99 in the original problem formulation. It turns out that the optimization of the modified problem formulation of Equations 96 – 99 and 117 on the boundary $\mathcal{B}$ of its feasible region $\mathcal{F}'$ is a more general result, as it is proven next.

Theorem 3 *Suppose that the optimization problem defined by Equations 96 – 99 and 117 has a non-empty feasible region $\mathcal{F}' \subseteq \mathcal{F}$. Then, it has an optimal solution located on the boundary $\mathcal{B}$ of $\mathcal{F}'$.*

Proof: Theorem 3 is trivially satisfied in the case that $\mathcal{B} = \mathcal{F}'$. Hence, in the following, suppose that the considered problem formulation has $\mathcal{B} \subset \mathcal{F}' \subseteq \mathcal{F} = [0, 1]^{dim(\omega)}$, and that $\omega^* \in \mathcal{F}' \backslash \mathcal{B}$ is an optimal solution for it. Then, it follows that there exists a sufficiently small $\delta > 0$, such that the entire neighborhood $N(\omega^*, \delta) \subseteq \mathcal{F}' \subseteq \mathcal{F}$. Moreover, the results of Section 6.2 regarding the monotonicity of the function $TH(\omega)$ with respect to the various control variables ω_{ij}, imply that there exists a *non-deteriorating* direction from ω^* to another point ω_1, located on the surface of the sphere $N(\omega^*, \delta)$. In case that the problem has a unique optimal solution, the above argument implies that $TH(\omega_1) > TH(\omega^*)$, which contradicts the original assumption that ω^* is an *optimal interior* point, and establishes the validity of the theorem.

To address the case of the existence of many optimal solutions, notice that the structure of the throughput function implied by Equation 106, combined with the existence of an optimal interior point of $\mathcal{F}'$, imply that $TH(\omega)$ is constant over $\mathcal{F}'$, and therefore, the entire set $\mathcal{B}$ constitutes a (sub-)set of optimal solutions. $\diamondsuit$

[4] and as it can also be seen in the above example...

A critical reading of the proof of Theorem 3 reveals that: (i) it is an immediate consequence of the structure of the $TH(\omega; \bar{q}_{ij})$ function, established by Proposition 1, and the implied monotonicity properties; (ii) it would apply to any other set of constraints appended to the original formulation of Equation 96 – 99(i.e., other than Constraint 117). Its availability provides an analytical insight that can be useful in the design of solution algorithms for any such extended formulations, by taking advantage of the particular form / structure of the appended constraint sets, and / or the establishment of further analytical / qualitative results for the resulting optimization problem(s). This has been, for instance, the case regarding the structure of the optimal scheduling policy, discussed in Chapter 3.

CHAPTER 7

MAJOR CONTRIBUTIONS AND SUGGESTED EXTENSIONS OF THIS WORK

In this chapter, we conclude the thesis by highlighting the major contributions and suggesting potential extensions of this work.

7.1 The Major Contributions

This work has developed an analytical framework and a practical basis for addressing the scheduling problem in the capacitated re-entrant line. The objective was to (i) develop an analytical framework for modeling, analysis and control of capacitated re-entrant line, (ii) provide a systematic and computationally effective method for computing the optimal scheduling policy for any given CRL configuration, and (iii) suggest an efficient approximating scheme to the optimal scheduling policy, while maintaining computational tractability. The consideration of the finite buffering capacity into the re-entrant line scheduling problem introduced additional material flow dynamics, that necessitate the integration of performance-oriented control and structural control. In this context, the Generalized stochastic Petri net(GSPN) framework was employed because of its capability to represent timed and untimed dynamics of the system behavior separately and clearly. Starting from this framework, we were able to develop the corresponding CT-AR-MDP problem, which is instrumental for the subsequent design of a novel approximating scheme to the optimal scheduling policy, based on the emerging NDP theory. A more systematic exposition of the major contributions of this thesis research has as follows:

Generalized Stochastic Petri Net-based Performance Analysis and Control of Capacitated Re-entrant Lines In this part of research, a formal system representation for the capacitated re-entrant line, based on the framework of GSPN, was developed and it

was shown that it (i) allows the seamless integration of logical / structural and timed-based aspects of the system behavior, (ii) provides an analytical formulation for the underlying scheduling problem, and (iii) leads to some interesting qualitative insights regarding the structure of the optimal scheduling policy. For the case of small-sized system, the proposed framework supported the thorough characterization of the structure of the optimal policies under various system parameterizations, allowing for a more systematic study and a more profound understanding of the dynamics taking place in these environments.

Markov Decision Process-based Performance Analysis and Control of Capacitated Re-entrant Lines The results of the previous part were subsequently utilized towards the development of a systematic and computationally effective method for computing the optimal scheduling policy for any given CRL configuration by (i) transforming the underlying scheduling problem to a MDP problem, and (ii) developing an algorithm that systematically generates the MDP formulation for any given fab configuration, while leading to a substantial reduction of the underlying state space. Even though this approach remained too complex for practical implementation, it provided further qualitative insights and a benchmarking baseline for the design of an efficient and computationally tractable approximating scheme to the optimal scheduling policy.

An Experimental Investigation of Feature-based Relative Value Function Approximation for Performance Control of Capacitated Re-entrant Lines This part of the work addressed the development of a near-optimal and computationally tractable approximating scheme to the optimal policy of the CRL scheduling problem by using a basis of "feature" functions to obtain a compact representation of the optimal relative value function. A set Φ of feature functions was suggested, and the quality of the generated approximation was systematically assessed through a numerical experiment. The efficacy and robustness of the proposed approximating scheme were also investigated through appropriate "what-if" / sensitivity analyses.

Optimal Deadlock Resolution in Resource Allocation Systems The last part of the presented research considered the analytical characterization of the optimality of various deadlock resolution strategies for Markovian resource allocation systems, under the objective of maximizing throughput. The alternative deadlock resolution strategies were formally characterized in the FSA context modeling the RAS behavior, and the problem of determining the optimal deadlock resolution strategy was reduced to the problem of identifying an optimal randomizing policy on an induced CT-MC. It was also shown that in certain cases, there will exist an optimal deadlock resolution strategy with a deterministic structure.

7.2 Suggested Extensions

The following topics are important extensions of the results reported in Chapter 5. Their systematic investigation would exceed, by far, the scope of a single Ph.D. thesis, but their resolution is necessary for the full-fledged development of the approximation method proposed in this work.

Development of a tuning algorithm that is computationally tractable for large-scale CRL configurations In Chapter 5, the potential capability of the NDP-based approximating scheme for the CRL scheduling problem was investigated through a numerical experiment employing small-sized re-entrant lines, and a set Φ of feature functions was suggested. We notice, however, that the application of this approximating scheme to large-scale CRL configurations will necessitate the development of a tuning algorithm for obtaining the weight vector r^*. The development of such an algorithm can be based on the combination of some other DP algorithms with simulation; *approximate policy iteration* [5] is such a typically used method. However, the overall convergence of such a computational scheme, for the average reward MDP problem, has been a challenging issue and needs to be systematically addressed in the considered application context.

Investigating the quality of the approximations obtained through the employment of distance metrics other than the l_∞-norm In this work, the quality of the

approximation provided by the suggested feature set was evaluated based on the l_∞-norm. However, it could be useful to consider other performance / distance metrics representing the "goodness-of-fit", such as the l_2-norm, and investigate the possibility of potential performance improvement.

A statistical assessment of the *significance* of the various feature functions for the performance of the resulting approximation The considered approximating scheme suggested sets of feature functions consisting of simple features, characterizing the distribution of the running process instances to the various processing stages, and their lower (2nd or 3rd) order interactions. We notice, however, that the cardinality of those sets of feature functions affects the computational complexity of the approximating scheme, and the convergence of the aforementioned tuning algorithm. Therefore, it is important to identify a "*minimal*" "good" subset of those feature functions, by systematically assessing their statistical significance on the performance of the resulting approximation.

Extension of the developed results to more general RAS classes Finally, it is useful to extend the developed results to more general RAS classes by considering additional operational requirements, such as the requirement for production according to a specified product mix. The introduction of such additional requirements in the underlying problem formulation takes us into the realm of constrained MDPs and raises the interesting problem of extending the basic NDP theory to accommodate these additional problem elements.

APPENDIX A

SENSITIVITY ANALYSIS OF THE RANDOMIZED POLICY GENERATED BY USING Φ WITH RESPECT TO THE PARAMETER VECTOR (δ, w)

Appendix A presents the results of a numerical experiment for showing the variation of the performance of the randomized policy generated by using Φ, as a function of the parameter vector (δ, w).

A.1 A capacitated re-entrant line consisting of 2 single-server workstations

Tables 19 – 21 characterize the performance of the randomized policy generated by using Φ, for Configurations 1 – 3.

Table 19: Performance of the randomized policy generated by using Φ for Conf 1

δ	w	Best w	Avg. % error	Min. % error	Max. % error	Avg. # of additional controls per state
0		0.99	0.093051	0.016866	0.173766	0
0.001		0.99	0.093051	0.016866	0.173766	0
0.002		0.99	0.093051	0.016866	0.173766	0
0.003		0.99	0.093051	0.016866	0.173766	0
0.004		0.99	0.229818	0.016866	1.595483	0.011111
0.005		0.99	0.229818	0.016866	1.595483	0.011111
0.006		0.99	0.229818	0.016866	1.595483	0.011111
0.007		0.99	0.229818	0.016866	1.595483	0.011111
0.008		0.99	0.269759	0.016866	1.595483	0.014815
0.009		0.99	0.269759	0.016866	1.595483	0.014815
0.010		0.99	0.269759	0.016866	1.595483	0.014815
0.011		0.99	0.269759	0.016866	1.595483	0.014815
0.012		0.99	0.494210	0.016866	6.907309	0.018519
0.013		0.99	0.494210	0.016866	6.907309	0.018519
0.014		0.99	0.610178	0.016866	6.907309	0.022222
0.015		0.99	0.628897	0.020716	6.907309	0.025926
0.016		0.99	0.628897	0.020716	6.907309	0.025926
0.017		0.99	0.854979	0.020716	7.175511	0.037037
0.018		0.99	0.854979	0.020716	7.175511	0.037037
0.019		0.99	1.116365	0.020716	7.175511	0.044444
0.020		0.99	1.116365	0.020716	7.175511	0.044444

Table 20: Performance of the randomized policy generated by using Φ for Conf 2

δ	w	Best w	Avg. % error	Min. % error	Max. % error	Avg. # of additional controls per state
0	0.80 - 0.99	0.99	0.735309	0.001030	5.127699	0.053333
0.001	0.80 - 0.99	0.98	0.820087	0.003095	4.886354	0.079524
0.002	0.80 - 0.99	0.98	0.868923	0.004468	4.886354	0.109524
0.003	0.80 - 0.99	0.98	0.865185	0.004468	4.886354	0.150952
0.004	0.80 - 0.99	0.98	0.835588	0.004989	4.886354	0.167143
0.005	0.80 - 0.99	0.98	0.854225	0.010308	4.886354	0.192857
0.006	0.80 - 0.99	0.98	0.874862	0.011343	4.882201	0.220000
0.007	0.80 - 0.99	0.99	0.876411	0.010995	5.125841	0.243333
0.008	0.80 - 0.99	0.99	0.904169	0.010995	5.125841	0.259524
0.009	0.80 - 0.99	0.99	0.945013	0.014776	5.127810	0.278095
0.010	0.80 - 0.99	0.99	0.954904	0.014776	5.127810	0.296191
0.011	0.80 - 0.99	0.99	0.977600	0.017020	5.397267	0.313333
0.012	0.80 - 0.99	0.99	0.970628	0.017020	5.397267	0.331905
0.013	0.80 - 0.99	0.99	0.984534	0.017689	5.397267	0.348571
0.014	0.80 - 0.99	0.99	1.005118	0.017552	5.397267	0.363333
0.015	0.81 - 0.99	0.99	1.011306	0.017829	5.481947	0.377143
0.016	0.81 - 0.99	0.99	1.036916	0.017829	5.481947	0.395238
0.017	0.82 - 0.99	0.99	1.050680	0.012706	5.481947	0.418095
0.018	0.83 - 0.99	0.99	1.183877	0.012706	5.498028	0.436191
0.019	0.83 - 0.99	0.99	1.184720	0.011854	5.498028	0.446667
0.020	0.84 - 0.99	0.99	1.256385	0.022849	7.584006	0.462381

Table 21: Performance of the randomized policy generated by using Φ for Conf 3

δ	w	Best w	Avg. % error	Min. % error	Max. % error	Avg. # of additional controls per state
0	0.90 - 0.99	0.99	0.759433	0.000104	4.078975	0.004364
0.001	0.91 - 0.99	0.99	0.794085	0.000104	4.477997	0.094909
0.002	0.92 - 0.99	0.99	0.795309	0.000104	4.491220	0.161939
0.003	0.92 - 0.99	0.99	0.798720	0.000104	4.252134	0.221818
0.004	0.92 - 0.99	0.99	0.799881	0.000104	4.501679	0.279394
0.005	0.93 - 0.99	0.99	0.810176	0.000104	4.661001	0.341818
0.006	0.93 - 0.99	0.99	0.754133	0.000104	4.467233	0.394303
0.007	0.93 - 0.99	0.99	0.783773	0.000104	4.385853	0.446545
0.008	0.92 - 0.99	0.99	0.740703	0.000104	4.298471	0.494909
0.009	0.92 - 0.99	0.99	0.726640	0.000104	4.365615	0.544848
0.010	0.92 - 0.99	0.99	0.705130	0.000104	4.266966	0.592364
0.011	0.93 - 0.99	0.99	0.714999	0.000104	3.523133	0.634667
0.012	0.94 - 0.99	0.99	0.762868	0.000104	3.564595	0.673697
0.013	0.97 - 0.99	0.99	0.935125	0.000104	5.069464	0.712606
0.014	0.99	0.99	1.005732	0.000104	5.061678	0.750424
0.015		0.99	1.241929	0.000104	6.374007	0.782061
0.016		0.99	1.321584	0.000104	6.374763	0.814182
0.017		0.99	1.382009	0.000104	6.358428	0.843394
0.018		0.99	1.382612	0.000104	6.361197	0.872848
0.019		0.99	1.439296	0.000104	6.358181	0.901697
0.020		0.99	1.464227	0.000104	6.371747	0.929333

A.2 A capacitated re-entrant line consisting of 3 single-server workstations

Tables 22 – 24 characterize the performance of the randomized policy generated by using Φ, for Configurations 4 – 6.

Table 22: Performance of the randomized policy generated by using Φ for Conf 4

δ	w	Best w	Avg. % error	Min. % error	Max. % error	Avg. # of additional controls per state
0	0.97 - 0.99	0.99	0.534733	0.034593	3.041891	0
0.001	0.96 - 0.99	0.99	0.477126	0.037824	3.041891	0.003529
0.002	0.97 - 0.99	0.99	0.492820	0.065391	2.837509	0.013333
0.003	0.98 - 0.99	0.99	0.545066	0.065391	2.837509	0.019216
0.004	0.98 - 0.99	0.99	0.525297	0.065391	1.908183	0.025490
0.005	0.98 - 0.99	0.99	0.542008	0.065391	1.908183	0.030588
0.006	0.99	0.99	0.560063	0.065672	1.908183	0.036471
0.007		0.99	0.614346	0.065672	2.019110	0.039608
0.008		0.99	0.641848	0.113849	2.042185	0.045882
0.009		0.99	0.692671	0.113849	2.241936	0.050980
0.010		0.99	0.691980	0.113849	2.159015	0.057255
0.011		0.99	0.687502	0.113849	2.162996	0.061961
0.012		0.99	0.700074	0.113849	2.338436	0.067843
0.013		0.99	0.750751	0.113849	2.532873	0.075294
0.014		0.99	0.903921	0.142047	2.773141	0.082353
0.015		0.99	1.013968	0.142047	3.097869	0.088627
0.016		0.99	1.154134	0.142047	4.191957	0.094118
0.017		0.99	1.264722	0.142047	5.255995	0.100784
0.018		0.99	1.300057	0.142047	5.584523	0.105882
0.019		0.99	1.383950	0.142047	5.860293	0.112549
0.020		0.99	1.404588	0.142047	5.860293	0.120392

Table 23: Performance of the randomized policy generated by using Φ for Conf 5

δ	w	Best w	Avg. % error	Min. % error	Max. % error	Avg. # of additional controls per state
0	0.80 - 0.99	0.99	0.848183	0.000522	3.796421	0.000507
0.001	0.80 - 0.99	0.99	0.830854	0.000344	3.796977	0.037246
0.002	0.80 - 0.99	0.99	0.789731	0.001920	3.826330	0.076957
0.003	0.80 - 0.99	0.99	0.751404	0.003151	3.830637	0.111087
0.004	0.80 - 0.99	0.99	0.723601	0.003318	2.617387	0.144638
0.005	0.80 - 0.99	0.99	0.727742	0.002974	2.618117	0.178478
0.006	0.80 - 0.99	0.99	0.724251	0.002797	2.597629	0.207101
0.007	0.80 - 0.99	0.99	0.760155	0.003851	2.615097	0.236232
0.008	0.80 - 0.99	0.99	0.760384	0.005072	2.241364	0.267609
0.009	0.80 - 0.99	0.99	0.807055	0.006126	2.243915	0.293985
0.010	0.80 - 0.99	0.99	0.808570	0.017678	2.414564	0.320145
0.011	0.80 - 0.99	0.99	0.896089	0.018199	2.467820	0.344348
0.012	0.80 - 0.99	0.99	0.953200	0.018199	2.496924	0.366739
0.013	0.80 - 0.99	0.99	0.997703	0.032020	2.612905	0.388696
0.014	0.81 - 0.99	0.99	1.019512	0.033318	2.662778	0.408623
0.015	0.82 - 0.99	0.99	1.065643	0.033129	3.270574	0.427899
0.016	0.84 - 0.99	0.99	1.137740	0.033129	3.485307	0.444928
0.017	0.85 - 0.99	0.99	1.200239	0.033041	3.985306	0.462899
0.018	0.87 - 0.99	0.99	1.336001	0.032209	4.005286	0.480000
0.019	0.89 - 0.99	0.99	1.409442	0.032297	4.097952	0.496812
0.020	0.90 - 0.99	0.99	1.495321	0.032209	4.286462	0.513551

Table 24: Performance of the randomized policy generated by using Φ for Conf 6

δ	w	Best w	Avg. % error	Min. % error	Max. % error	Avg. # of additional controls per state
0	0.80 - 0.99	0.98	0.808917	0.002273	4.734076	0.000031
0.001	0.80 - 0.99	0.99	0.802225	0.000376	4.626198	0.051220
0.002	0.80 - 0.99	0.99	0.780900	0.000248	3.672201	0.096725
0.003	0.80 - 0.99	0.99	0.793988	0.000880	3.585386	0.142570
0.004	0.80 - 0.99	0.99	0.802864	0.000753	3.585446	0.181312
0.005	0.80 - 0.99	0.99	0.775288	0.000504	3.848678	0.226290
0.006	0.80 - 0.99	0.99	0.767967	0.000625	3.862528	0.266728
0.007	0.80 - 0.99	0.99	0.727640	0.000625	3.502581	0.303182
0.008	0.80 - 0.99	0.99	0.752857	0.000880	3.547801	0.338214
0.009	0.82 - 0.99	0.99	0.832796	0.001008	3.707656	0.370312
0.010	0.85 - 0.99	0.99	0.901164	0.003025	4.713343	0.398146
0.011	0.87 - 0.99	0.99	0.956886	0.011364	5.178113	0.424838
0.012	0.91 - 0.99	0.99	1.072974	0.010348	5.429686	0.448656
0.013	0.95 - 0.99	0.99	1.186413	0.013004	5.585717	0.471671
0.014	0.98 - 0.99	0.99	1.247086	0.014233	6.029005	0.495860
0.015		0.99	1.288878	0.014233	6.093718	0.518072
0.016		0.99	1.350094	0.014233	6.340556	0.537349
0.017		0.99	1.414435	0.015654	6.481371	0.556472
0.018		0.99	1.442196	0.015890	6.516797	0.573957
0.019		0.99	1.470046	0.015473	6.529207	0.590516
0.020		0.99	1.533689	0.015834	6.542250	0.605005

APPENDIX B

SENSITIVITY ANALYSIS OF THE RANDOMIZED POLICY GENERATED BY USING Φ WITH RESPECT TO THE PARAMETER VECTOR (δ, w) FOR LARGE-SIZED RE-ENTRANT LINES

Appendix B presents the results of a numerical experiment that investigates the sensitivity of the randomized policy generated by using Φ with respect to the parameter vector (δ, w) for the case of "large-sized" re-entrant lines.

B.1 A capacitated re-entrant line generated by increasing the buffering capacity

Tables 25 and 26 characterize the performance of the randomized policy generated by using Φ, for Configurations 7 and 8.

Table 25: Performance of the randomized policy generated by using Φ for Conf 7

δ	w	Best w	Avg. % error	Min. % error	Max. % error	Avg. # of additional controls per state
0	0.92 - 0.99	0.99	0.116272	0.000603	0.550340	0.000646
0.001	0.93 - 0.99	0.99	0.140766	0.000603	0.550091	0.066865
0.002	0.94 - 0.99	0.99	0.153714	0.000662	0.554420	0.126343
0.003	0.93 - 0.99	0.99	0.137892	0.000603	0.569249	0.181534
0.004	0.95 - 0.99	0.99	0.195810	0.000439	0.563325	0.243853
0.005	0.95 - 0.99	0.99	0.234524	0.000766	1.001620	0.307128
0.006	0.95 - 0.99	0.99	0.229157	0.000766	0.702816	0.367562
0.007	0.95 - 0.99	0.99	0.248541	0.000993	0.938772	0.427195
0.008	0.95 - 0.99	0.99	0.236167	0.001083	0.792322	0.481792
0.009	0.96 - 0.99	0.99	0.316179	0.001162	1.063212	0.534814
0.010	0.98 - 0.99	0.99	0.451078	0.001162	2.416435	0.587707
0.011		0.99	0.573790	0.001123	2.897800	0.641090
0.012		0.99	0.638728	0.001123	2.960590	0.695015
0.013		0.99	0.649719	0.001202	3.042663	0.745325
0.014		0.99	0.623865	0.001202	3.051514	0.797882
0.015		0.99	0.682222	0.000874	2.842988	0.845067
0.016		0.99	0.680100	0.000874	2.817514	0.893879
0.017		0.99	0.719243	0.000785	2.836415	0.941994
0.018		0.99	0.690551	0.000745	2.407415	0.987732
0.019		0.99	0.731725	0.000745	2.326482	1.033316
0.020		0.99	0.717120	0.000745	2.327077	1.078048

Table 26: Performance of the randomized policy generated by using Φ for Conf 8

δ	w	Best w	Avg. % error	Min. % error	Max. % error	Avg. # of additional controls per state
0	0.88 - 0.99	0.97	0.078278	0.001046	0.175952	0.000020
0.001	0.88 - 0.99	0.98	0.074681	0.000395	0.183286	0.061409
0.002	0.91 - 0.99	0.99	0.085371	0.000264	0.214044	0.120783
0.003	0.93 - 0.99	0.99	0.084560	0.000132	0.200732	0.179517
0.004	0.94 - 0.99	0.99	0.089720	0.000264	0.240284	0.237413
0.005	0.97 - 0.99	0.99	0.122927	0.000264	0.527734	0.294899
0.006		0.99	0.184087	0.000264	0.612727	0.352426
0.007		0.99	0.299548	0.000264	1.742189	0.410621
0.008		0.99	0.442921	0.000264	3.118760	0.462368
0.009		0.99	0.703544	0.000179	5.049879	0.513316
0.010		0.99	0.810791	0.000264	5.561761	0.563406
0.011		0.99	0.883834	0.000264	5.768345	0.612468
0.012		0.99	0.927501	0.000264	5.913092	0.657926
0.013		0.99	0.986787	0.000264	5.980769	0.702625
0.014		0.99	1.049240	0.000264	6.089511	0.743691
0.015		0.99	1.145852	0.000264	6.201651	0.785716
0.016		0.99	1.187435	0.000359	6.226059	0.826223
0.017		0.99	1.205351	0.000359	6.257959	0.863336
0.018		0.99	1.226184	0.000264	6.270335	0.897914
0.019		0.99	1.245896	0.000359	6.279448	0.931643
0.020		0.99	1.257809	0.000359	6.293310	0.963346

B.2 A capacitated re-entrant line generated by increasing the number of workstations and job stages

Table 27 characterizes the performance of the randomized policy generated by using Φ, for Configuration 9.

Table 27: Performance of the randomized policy generated by using Φ for Conf 9

δ	Avg. % error	Min. % error	Max. % error	Avg. # of additional controls per state
0	3.550095	0.188082	7.519532	0
0.001	3.549228	0.197071	7.473375	0.021690
0.002	3.555097	0.197335	7.497574	0.045382
0.003	3.600749	0.231355	7.523381	0.069751
0.004	3.551931	0.230181	7.264050	0.093040
0.005	3.567805	0.234742	7.199137	0.116520
0.006	3.577250	0.229133	7.196691	0.139643
0.007	3.743047	0.285441	7.695214	0.164391
0.008	3.794992	0.284656	7.765227	0.189174
0.009	3.753479	0.317767	7.734190	0.215901
0.010	3.758613	0.313331	7.641400	0.243007
0.011	3.774743	0.339139	7.649339	0.270350
0.012	3.830793	0.344484	7.753351	0.296402
0.013	3.870389	0.343311	7.726382	0.323377
0.014	3.869953	0.343963	7.713537	0.350958
0.015	3.940649	0.390102	7.671853	0.378775
0.016	3.913728	0.346574	7.551903	0.407007
0.017	3.888128	0.367293	7.347809	0.434694
0.018	3.926348	0.368598	7.391554	0.462227
0.019	3.994647	0.407046	7.453629	0.489179
0.020	3.969254	0.434027	7.212045	0.517767

APPENDIX C

SENSITIVITY ANALYSIS OF THE RANDOMIZED POLICY GENERATED BY USING Φ' WITH RESPECT TO THE PARAMETER VECTOR (δ, w)

Appendix C presents the results of a numerical experiment for showing the variation of the performance of the randomized policy generated by using Φ', as a function of the parameter vector (δ, w).

C.1 A capacitated re-entrant line consisting of 2 single-server workstations

Tables 28 – 30 characterize the performance of the randomized policy generated by using Φ', for Configurations 1 – 3.

Table 28: Performance of the randomized policy generated by using Φ' for Conf 1

δ	w	Best w	Avg. % error	Min. % error	Max. % error	Avg. # of additional controls per state
0		0.99	0.093051	0.016866	0.173766	0
0.001		0.99	0.093051	0.016866	0.173766	0
0.002		0.99	0.093051	0.016866	0.173766	0
0.003		0.99	0.093051	0.016866	0.173766	0
0.004		0.99	0.093051	0.016866	0.173766	0
0.005		0.99	0.093051	0.016866	0.173766	0
0.006		0.99	0.093051	0.016866	0.173766	0
0.007		0.99	0.093051	0.016866	0.173766	0
0.008		0.99	0.093051	0.016866	0.173766	0
0.009		0.99	0.093051	0.016866	0.173766	0
0.010		0.99	0.093051	0.016866	0.173766	0
0.011		0.99	0.093051	0.016866	0.173766	0
0.012		0.99	0.093051	0.016866	0.173766	0
0.013		0.99	0.093051	0.016866	0.173766	0
0.014		0.99	0.093051	0.016866	0.173766	0
0.015		0.99	0.093051	0.016866	0.173766	0
0.016		0.99	0.093051	0.016866	0.173766	0
0.017		0.99	0.093051	0.016866	0.173766	0
0.018		0.99	0.093051	0.016866	0.173766	0
0.019		0.99	0.093051	0.016866	0.173766	0
0.020		0.99	0.093051	0.016866	0.173766	0

Table 29: Performance of the randomized policy generated by using Φ' for Conf 2

δ	w	Best w	Avg. % error	Min. % error	Max. % error	Avg. # of additional controls per state
0	0.80 - 0.99	0.99	0.067502	0.000828	0.215701	0.412857
0.001	0.80 - 0.99	0.99	0.084444	0.017280	0.215701	0.529524
0.002	0.80 - 0.99	0.99	0.109094	0.017280	0.340854	0.570952
0.003	0.80 - 0.99	0.99	0.129176	0.017280	0.356791	0.593809
0.004	0.80 - 0.99	0.99	0.156615	0.017280	0.772878	0.618095
0.005	0.80 - 0.99	0.99	0.174581	0.017280	0.772878	0.634286
0.006	0.80 - 0.99	0.99	0.209774	0.021602	0.773312	0.662381
0.007	0.80 - 0.99	0.99	0.227573	0.021602	0.773312	0.673809
0.008	0.80 - 0.99	0.99	0.244239	0.021602	0.773312	0.683809
0.009	0.80 - 0.99	0.99	0.256469	0.021602	0.773312	0.695238
0.010	0.80 - 0.99	0.99	0.273782	0.021602	0.773312	0.709048
0.011	0.80 - 0.99	0.99	0.304110	0.021602	0.773915	0.725714
0.012	0.80 - 0.99	0.99	0.323499	0.021602	1.015975	0.734286
0.013	0.80 - 0.99	0.99	0.391199	0.021602	1.015975	0.748095
0.014	0.81 - 0.99	0.99	0.562778	0.021602	2.805096	0.763810
0.015	0.81 - 0.99	0.99	0.595602	0.021602	2.805096	0.777619
0.016	0.81 - 0.99	0.99	0.608435	0.024066	2.805096	0.787619
0.017	0.82 - 0.99	0.99	0.620441	0.024066	2.805096	0.796191
0.018	0.82 - 0.99	0.99	0.681371	0.044514	2.805096	0.806191
0.019	0.82 - 0.99	0.99	0.700402	0.044514	2.805096	0.815238
0.020	0.83 - 0.99	0.99	0.778178	0.044514	3.159263	0.827143

Table 30: Performance of the randomized policy generated by using Φ' for Conf 3

δ	w	Best w	Avg. % error	Min. % error	Max. % error	Avg. # of additional controls per state
0	0.81 - 0.99	0.98	0.420034	0.000305	3.184219	0.042424
0.001	0.82 - 0.99	0.98	0.440685	0.001138	3.099168	0.303030
0.002	0.83 - 0.99	0.98	0.447695	0.001361	3.093156	0.375273
0.003	0.83 - 0.99	0.98	0.441527	0.000684	3.103770	0.432606
0.004	0.83 - 0.99	0.98	0.436784	0.000684	3.103770	0.489697
0.005	0.84 - 0.99	0.98	0.496701	0.000908	3.137010	0.540000
0.006	0.85 - 0.99	0.98	0.516987	0.000908	3.148724	0.589818
0.007	0.90 - 0.99	0.98	0.813306	0.000908	5.055142	0.627879
0.008	0.92 - 0.99	0.99	0.883697	0.000684	6.351899	0.671515
0.009	0.93 - 0.99	0.98	0.893463	0.000454	6.357926	0.711273
0.010	0.93 - 0.99	0.98	0.902405	0.000684	6.602105	0.746061
0.011	0.93 - 0.99	0.99	0.902353	0.000684	6.601101	0.779636
0.012	0.93 - 0.99	0.99	0.913397	0.000684	6.643806	0.806909
0.013	0.94 - 0.99	0.99	0.911973	0.000908	6.643806	0.836000
0.014	0.94 - 0.99	0.99	0.908630	0.000908	6.643806	0.866667
0.015	0.95 - 0.99	0.99	0.917909	0.000908	6.650843	0.894667
0.016	0.97 - 0.99	0.99	0.967995	0.000908	6.650843	0.921818
0.017	0.99	0.99	1.011875	0.000908	6.651846	0.943151
0.018	0.98 - 0.99	0.99	0.985634	0.000908	6.651846	0.961333
0.019		0.99	1.065495	0.000908	6.651846	0.985212
0.020		0.99	1.054144	0.000908	6.651846	1.006667

C.2 A capacitated re-entrant line consisting of 3 single-server workstations

Tables 31 – 33 characterize the performance of the randomized policy generated by using Φ', for Configurations 4 – 6.

Table 31: Performance of the randomized policy generated by using Φ' for Conf 4

δ	w	Best w	Avg. % error	Min. % error	Max. % error	Avg. # of additional controls per state
0	0.89 - 0.99	0.99	0.046410	0.001023	0.193298	0
0.001	0.89 - 0.99	0.99	0.054492	0.002381	0.193298	0.009020
0.002	0.90 - 0.99	0.99	0.097059	0.005842	0.542286	0.019608
0.003	0.91 - 0.99	0.99	0.148483	0.005842	0.731920	0.034118
0.004	0.92 - 0.99	0.99	0.194727	0.005842	1.310337	0.043137
0.005	0.93 - 0.99	0.99	0.254006	0.005842	1.323806	0.052157
0.006	0.95 - 0.99	0.99	0.340700	0.005842	1.687384	0.060392
0.007	0.96 - 0.99	0.99	0.418782	0.011753	1.706932	0.072549
0.008	0.97 - 0.99	0.99	0.442019	0.011753	1.717618	0.078431
0.009	0.99	0.99	0.559924	0.011753	3.390005	0.084314
0.010		0.99	0.714375	0.011753	4.086698	0.095294
0.011		0.99	0.905044	0.011753	5.345923	0.106275
0.012		0.99	0.939691	0.011753	5.345923	0.111373
0.013		0.99	1.030909	0.011753	6.934800	0.116471
0.014		0.99	1.081743	0.015260	6.934800	0.121961
0.015		0.99	1.147258	0.015260	6.934800	0.127843
0.016		0.99	1.203572	0.015260	6.972301	0.136471
0.017		0.99	1.238225	0.015260	6.985514	0.141961
0.018		0.99	1.385380	0.015260	6.985514	0.152941
0.019		0.99	1.415286	0.015260	6.985514	0.155686
0.020		0.99	1.548041	0.015260	8.035472	0.163922

Table 32: Performance of the randomized policy generated by using Φ' for Conf 5

δ	w	Best w	Avg. % error	Min. % error	Max. % error	Avg. # of additional controls per state
0	0.80 - 0.99	0.99	0.301631	0.001044	1.250065	0.009855
0.001	0.80 - 0.99	0.99	0.281207	0.000344	1.106929	0.080000
0.002	0.80 - 0.99	0.99	0.236877	0.005249	0.692937	0.130290
0.003	0.80 - 0.99	0.99	0.253872	0.005249	0.789117	0.176232
0.004	0.80 - 0.99	0.99	0.285927	0.007002	0.821935	0.214710
0.005	0.80 - 0.99	0.99	0.308379	0.008578	0.826193	0.249493
0.006	0.80 - 0.99	0.99	0.336819	0.008578	1.089393	0.284275
0.007	0.80 - 0.99	0.99	0.358319	0.008578	1.097167	0.314348
0.008	0.80 - 0.99	0.99	0.375408	0.008578	1.095074	0.340797
0.009	0.80 - 0.99	0.99	0.397074	0.008578	1.139246	0.363551
0.010	0.80 - 0.99	0.99	0.425006	0.016942	1.139704	0.381304
0.011	0.80 - 0.99	0.99	0.636587	0.026808	3.494548	0.401884
0.012	0.80 - 0.99	0.99	0.836752	0.029779	4.583511	0.421449
0.013	0.82 - 0.99	0.99	1.021942	0.026994	5.806634	0.440290
0.014	0.84 - 0.99	0.99	1.186513	0.026805	7.829068	0.461449
0.015	0.86 - 0.99	0.99	1.298411	0.026994	7.931677	0.476739
0.016	0.87 - 0.99	0.99	1.370590	0.026428	7.931677	0.493623
0.017	0.88 - 0.99	0.99	1.461784	0.026528	8.210983	0.510362
0.018	0.89 - 0.99	0.99	1.482023	0.026528	8.285177	0.523623
0.019	0.90 - 0.99	0.99	1.522163	0.027083	8.836941	0.535652
0.020	0.91 - 0.99	0.99	1.575743	0.045423	8.932685	0.547391

Table 33: Performance of the randomized policy generated by using Φ' for Conf 6

δ	w	Best w	Avg. % error	Min. % error	Max. % error	Avg. # of additional controls per state
0	0.80 - 0.99	0.99	0.534215	0.000093	3.363043	0.001483
0.001	0.80 - 0.99	0.99	0.545871	0.000490	3.366581	0.057708
0.002	0.80 - 0.99	0.99	0.509121	0.001136	3.426700	0.107970
0.003	0.80 - 0.99	0.99	0.491644	0.002053	3.402330	0.156256
0.004	0.80 - 0.99	0.99	0.544619	0.002053	4.360024	0.199969
0.005	0.80 - 0.99	0.99	0.576663	0.002053	5.275200	0.237288
0.006	0.80 - 0.99	0.99	0.550321	0.002251	4.859274	0.272783
0.007	0.80 - 0.99	0.99	0.560726	0.002251	4.841228	0.306549
0.008	0.80 - 0.99	0.99	0.574190	0.008119	4.814780	0.336732
0.009	0.80 - 0.99	0.99	0.536298	0.007930	3.652459	0.366172
0.010	0.80 - 0.99	0.99	0.589486	0.008722	3.626673	0.393543
0.011	0.80 - 0.99	0.99	0.582649	0.008923	3.651802	0.420142
0.012	0.83 - 0.99	0.99	0.711390	0.009680	3.693819	0.442601
0.013	0.83 - 0.99	0.99	0.763298	0.009715	3.738007	0.462002
0.014	0.87 - 0.99	0.99	0.935176	0.003799	3.753684	0.480816
0.015	0.89 - 0.99	0.99	1.019545	0.004051	4.795834	0.500402
0.016	0.90 - 0.99	0.99	1.028562	0.003970	4.959922	0.517856
0.017	0.92 - 0.99	0.99	1.083575	0.004302	5.182458	0.534693
0.018	0.97 - 0.99	0.99	1.219053	0.004051	5.770144	0.550201
0.019	0.99	0.99	1.277318	0.003718	5.767331	0.564751
0.020		0.99	1.282736	0.005320	5.739575	0.578653

REFERENCES

[1] Ajmone Marsan, M., Balbo, G., and Conte, G., "A class of generalized stochastic petri nets for performance evaluation of multiprocessor systems," *ACM Trans. Comput. Sys.*, vol. 2, pp. 93–122, 1984.

[2] Ajmone Marsan, M., Balbo, G., and Conte, G., *Performance Models of Multiprocessor Systems*. Cambridge, MA: The MIT Press, 1986.

[3] Banaszak, Z. A. and Krogh, B. H., "Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows," *IEEE Trans. on Robotics and Automation*, vol. 6, pp. 724–734, 1990.

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

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

[6] Bertsimas, D., Paschalidis, I. C., and Tsitsiklis, J. N., "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.

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

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

[9] Choi, J. Y. and Reveliotis, S. A., "Relative value function approximation for capacitated re-entrant lines: An experimental investigation," Submitted to IEEE Conference on Decision and Control, 2004.

[10] Choi, J. Y. and Reveliotis, S. A., "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.

[11] Chvátal, V., *Linear Programming*. N.Y., N.Y.: W. H. Freeman & Co., 1983.

[12] Dai, J. G. and Lin, W., "Maximum pressure policies in stochastic processing networks," 2003. Preprint.

[13] de Farias, D. P., *The Linear Programming Approach to Approximate Dynamic Programming: Theory and Application*. PhD thesis, Massachusetts Institute of Technology, June 2002.

[14] Desrochers, A. A. and Al-Jaar, R. Y., *Applications of Petri nets in Manufacturing Systems*. Piscataway, NJ: IEEE Press, 1995.

[15] Garey, M. R. and Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY: W. H. Freeman and Co., 1979.

[16] Gershwin, S. B., "Hierarchical flow control: A framework for scheduling and planning discrete events in manufacturing systems," *Proceedings of the IEEE*, vol. 77, pp. 195–209, 1989.

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

[18] Guestrin, C., Koller, D., and Parr, R., "Max-norm projections for factored mdp's," in *International Joint Conference on Artificial Intelligence*, vol. 1, pp. 673–680, August 2001.

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

[20] Henderson, S. G., Meyn, S. P., and Tadic, V. B., "Performance evaluation and policy selection in multiclass networks," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 13, pp. 149–189, 2003.

[21] Jeng, M., Xie, X., and Hung, W. Y., "Markovian timed petri nets for performance analysis of semiconductor manufacturing systems," *IEEE Trans. on Systems, Man and Cybernetics - Part B*, vol. 30, pp. 757–771, 2000.

[22] Kumar, P. R., "Re-entrant lines," *Queueing Systems: Theory and Application, Special Issue on Queueing Networks*, vol. 13, pp. 87–110, 1993.

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

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

[25] Kumar, S. and Kumar, P. R., "The last buffer first serve priority policy is stable for stochastic re-entrant lines," tech. rep., Coordinated Science Laboratory, Univ. of Illinois, Urbana, IL, 1994.

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

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

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

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

[30] Lu, S. H. and Kumar, P. R., "Distributed scheduling based on due dates and buffer priorities," *IEEE Trans. on Aut. Control*, vol. 36, pp. 1406–1416, 1991.

[31] Lu, S. H., Ramaswamy, D., and Kumar, P. R., "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.

[32] Meyn, S., "Stability and optimization of queueing networks and their fluid models," in *Proceedings of the Summer Seminar on "The Mathematics of Stochastic Manufacturing Systems"* (Williamsburg, V., ed.), 1996.

[33] Meyn, S., "The policy iteration algorithm for average reward markov decision processes with general state space," *IEEE Trans. on Automatic Control*, vol. 42, pp. 1663–1680, 1997.

[34] Murata, T., "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, pp. 541–580, 1989.

[35] Nahmias, S., *Production and Operations Analysis – 3rd ed.* Chicago, IL: IRWIN, 1997.

[36] Papadopoulos, H. T., Heavy, C., and Browne, J., *Queueing Theory in Manufacturing Systems Analysis and Design*. New York, NY: Chapman & Hall, 1993.

[37] Park, J., Reveliotis, S. A., Bodner, D., Zhou, C., Wu, J.-F., and McGinnis, L., "High-fidelity rapid prototyping of 300mm fabs through discrete event system modeling," *Computers in Industry : invited paper for the special issue on MASM'2000*, vol. 45, pp. 79–98, 2001.

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

[39] Reveliotis, S. A., *Structural Analysis & Control of Flexible Manufacturing Systems with a Performance Perspective*. PhD thesis, University of Illinois, Urbana, IL, 1996.

[40] Reveliotis, S. A., "An analytical investigation of the deadlock avoidance vs. detection & recovery problem in buffer-space allocation of flexibly automated production systems," *IEEE Trans. on SMC: Part B*, vol. 30, pp. –, 2000.

[41] Reveliotis, S. A., "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.

[42] Reveliotis, S. A. and Choi, J. Y., "On the optimality of randomized deadlock avoidance policies," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 13, no. 4, pp. 303–320, 2003.

[43] Reveliotis, S. A. and Ferreira, P. M., "Deadlock avoidance policies for automated manufacturing cells," *IEEE Trans. on Robotics & Automation*, vol. 12, pp. 845–857, 1996.

[44] Reveliotis, S. A., Lawley, M. A., and Ferreira, P. M., "Polynomial complexity deadlock avoidance policies for sequential resource allocation systems," *IEEE Trans. on Automatic Control*, vol. 42, pp. 1344–1357, 1997.

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

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

[47] Sifakis, J., *Use of Petri nets for Performance Evaluation*. Amsterdam, Netherlands: North Holland, 1977.

[48] Singer, P., "The driving forces in cluster tool development," *Semiconductor International*, vol. July '95, pp. 113–118, 1995.

[49] Strang, G., *Linear Algebra and its Applications*. Harcourt College Pub., 3rd. ed. ed., 1988.

[50] Tesauro, G. J., "Practical issues in temporal-difference learning," *Machine Learning*, vol. 8, pp. 257–277, 1992.

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

[52] Tsitsiklis, J. N. and Van Roy, B., "Average cost temporal-difference learning," *Automatica*, vol. 35, no. 11, pp. 1799–1808, 1999.

[53] Van Roy, B., *Learning and Value Function Approximation in Complex Decision Processes*. PhD thesis, Massachusetts Institute of Technology, 1998.

[54] Van Roy, B., Bertsekas, D. P., Lee, Y., and Tsitsiklis, J. N., "A neuro-dynamic programming approach to retailer inventory management," in *Proceeding of the 36th IEEE Conference on Decision and Control*, pp. 4052–4057, December 1997.

[55] Viswanadlam, N. and Narahari, Y., *Performance Modeling of Automated Manufacturing Systems*. Englewood Cliffs, NJ: Prentice Hal, 1992.

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

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

[58] Zhang, W. and Dietterich, T. G., "A reinforcement learning approach to job shop scheduling," in *Proceedings of the IJCAI*, 1995.

[59] Zhou, M. and Jeng, M., "Modeling, analysis, simulation, scheduling and control of semiconductor manufacturing systems," *IEEE Trans. on Semiconductor Manufacturing*, vol. 11, pp. 333–357, 1998.

[60] Zuberek, W. M., "Timed petri nets in modeling and analysis of cluster tools," *IEEE Trans. on R&A*, vol. 17, pp. 562–575, 2001.

VITA

Jin Young Choi was born in Korea, on August 3, 1968. He received the B.S. degree in Industrial Engineering from Hanyang University, Seoul, Korea, in 1991 and the M.S. degree in Industrial Engineering from Korea Advanced Institute of Science and Technology(KAIST), Taejon, Korea, in 1993. From 1993 to 1999, he worked as a senior member of engineering staff at Electronics and Telecommunications Research Institute(ETRI), Korea, where he was involved in several projects and contributed to the development of highly advanced technologies for telecommunication networks. Since Fall, 1999, he has worked as a research assistant pursuing the Ph.D. degree in the School of Industrial & Systems Engineering at the Georgia Institute of Technology. His research interests include the area of Discrete Event Dynamic Systems theory, and its application to the analysis and design of control / information systems for manufacturing, logistics and telecommunication networks.