SCHEDULING TECHNIQUES FOR COMPLEX RESOURCE ALLOCATION SYSTEMS

A Dissertation
Presented to
The Academic Faculty

By

Michael Ibrahim

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Industrial Engineering

Georgia Institute of Technology

December 2019

SCHEDULING TECHNIQUES FOR COMPLEX RESOURCE ALLOCATION SYSTEMS

Approved by:

Dr. Spyros Reveliotis, Advisor
School of Industrial Engineering
Georgia Institute of Technology

Dr. Hayriye Ayhan
School of Industrial Engineering
Georgia Institute of Technology

Dr. Nagi Gebraeel
School of Industrial Engineering
Georgia Institute of Technology

Dr. Kamran Paynabar
School of Industrial Engineering
Georgia Institute of Technology

Dr. Douglas Down
Department of Computing and Software
McMaster University

Date Approved: August 15, 2019

To my family and friends.

ACKNOWLEDGMENTS

First and foremost, I would like to thank to my PhD advisor, Professor Spyros Reveliotis, for his patience, guidance, help and support during these past four years. Spyros is an excellent mentor and advisor, he is a thorough and rigorous researcher, and he helped me to improve my research skills. I could not imagine the completion of this work without his technical and editorial advising.

I would like to thank Professor Hayriye Ayhan, Professor Nagi Gebraeel, Professor Kamran Paynabar and Professor Douglas Down for serving on my thesis committee and providing valuable ideas and feedback. Also, I would like to thank Dr. Ahmed Nazeem for his collaboration, help and guidance. Finally, I would like to thank my fellow graduate students for making my stay in Atlanta much more pleasurable.

Last but not least, I would like to thank my sister and my mother for their care and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This research program provides a complete framework for the real-time management of complex sequential resource allocation systems (RAS) with blocking and deadlocking effects in their dynamics. This framework addresses both control objectives of logical correctness and performance optimization for the conisidered RAS.

A more detailed account of the thesis contributions is as follows:

For the logical-correctness part of the presented framework, we leverage some formal Discrete Event System (DES)-based representations of the RAS behavior and we introduce a new class of deadlock avoidance policies (DAPs) for the considered sequential RAS that is characterized as the class of "maximal linear" DAPs. We also provide a complete algorithm for enumerating all the elements of this policy class for a broad class of RAS instances. Finally, we present some numerical experimentation that demonstrates the efficacy of the presented algorithm.

For the performance-optimization part of the presented framework, we provide a scheduling methodology that aims to maximize the throughput of complex RAS with blocking and deadlocking effects. This methodology is based on the solution of a pertinent "fluid" relaxation of the addressed scheduling problem, and it is enabled by the pre-established ability to control the underlying RAS for deadlock freedom, and by the further ability to express the corresponding DAP as a set of linear inequalities on the system state.

Furthermore, we strengthen and further formalize these developments by taking advantage of the representational and analytical capabilities of the Petri net (PN) modeling framework, which is one of the main formal representational frameworks employed by the current DES theory. These capabilities enable a seamless treatment of the behavioral and the time-based dynamics of the underlying RAS, and they also support a notion of "fluidization" of these dynamics through the more recent developments in the area of timed and untimed continuous PN models; this last capability was especially critical for the system-

atic derivation of the sought "fluid relaxation" models and formulations. The information that is contained in the developed "fluid" models, when combined with the "linear" deadlock avoidance policies that have been employed in this work, provide a complete and very efficient controller for the considered RAS.

Finally, we present a "correction" algorithm that aims to detect potential suboptimal decisions that might be effected by the aforementioned controller and correct them. These "corrections" can be effected either in an "off-line" mode, by simulating the dynamics of the underlying RAS, or in an "on-line" mode where the underlying RAS is fully operational and the necessary corrections are inferred from the observed behavior of the system. In both of these modes, and especially the second one, the "correction" algorithm endows the developed control framework with a "learning" capability. From a more methodological standpoint, the results that enable this correcting mechanism are based on the sensitivity analysis of Markov reward processes and the statistical theory of "ranking & selection". A series of numerical results demonstrate and assess the efficacy of the developed methodology.

CHAPTER 1
INTRODUCTION

The research program presented in this document addresses the problem of providing a tractable methodology for scheduling complex workflows with finite reusable resources that take place in the context of repetitive and fully automated operations. The scheduling problems that are defined in these environments must address typical time-based performance objectives, like the maximization of the throughput of the underlying processes and/or the control of the experienced delays and congestions. But the finiteness of the resources in such an automated workflow, when combined with the complexity and the arbitrary structure of the underlying operations, can give rise to "blocking" and "deadlocking" effects. These deadlocks must be eliminated from the operation of the underlying workflow in order to ensure its uninterrupted operation. Furthermore, this deadlock elimination must be done in a way that will not impair dramatically the production capacity of the underlying system.

One "real-world" application that motivates this research is the contemporary manufacturing operations. For instance, in an automated manufacturing cell with a number of machines, different kinds of arriving parts may require processing by the machines in different but specified orders. On the other hand, each machine is capable of processing different part types, or the same part type at different stages, but one machine cannot process multiple parts at the same time. This flexibility gives rise to "competition" among the parts for the processing capacity of these machines and leads to a "scheduling" problem. In other words, when two or more parts, diversified either by their types or stages, are presented at the same machine, a decision must be made to exclusively allocate the machine to one of the parts in a way that optimizes some performance measure of interest.

Another application for workflow scheduling beyond the scope of the manufacturing

processes can be seen in the domain of internet-based workflow management for certain business processes [45]. Possible applications for the automation of business processes include the processing of loan applications, insurance claims, or passport applications. All these transactions follow well-established procedures and rely on human clerks, computer programs and other computational resources to carry out individual tasks. Also, all these transactions are repetitive and occur in an automated environment.

All the aforementioned applications share these common characteristics: (i) There exist a set of reusable but limited *resources*, such as the machines, human clerks, and the various computer applications and their supported resources. (ii) There exist a set of *processes*, such as the parts, and the loan applications, which require exclusive accessibility to some resources for a certain time span, in order to perform some tasks, such as the processing of the parts, or of the requested loans. (iii) The tasks of each process should be performed in a *sequential* manner, such as the "specified orders of the machines" for the parts, or the various transactional steps for loan processing. (iv) Finally, the resource requests from different tasks may overlap, but since the requested resources are limited, an *arbitration* is needed to assign *priorities* to the different tasks competing for the insufficient resources.

But as already mentioned, an additional characteristic of the previously cited examples is that all these systems are fully automated. For such systems, an additional task of the system "controller" is to maintain the smoothness and the logical correctness of the automated operation. For example, in a manufacturing cell, if the buffers of all the machines are full, but none of the parts is at its final stage to be eligible to exit the system, then no part can proceed further and the operation of the system becomes permanently stalled, or *deadlocked*. The operational policies aiming to prevent the formation of such states are typically called *logical* or *behavior control* in the relevant literature.

In order to optimize the performance of the complex resource allocation functions that have been discussed in the previous paragraph while maintaining logical correctness, the formal abstraction of the *sequential resource allocation system (RAS)* [63] was introduced

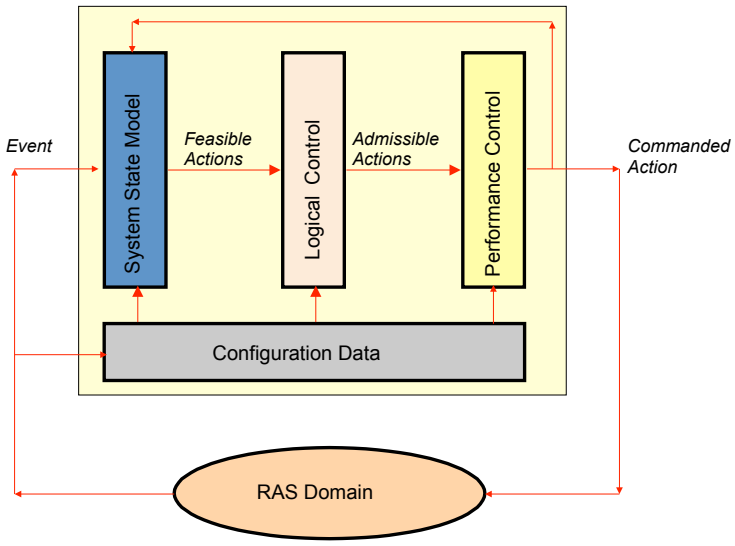

Figure 1.1: An event-driven control scheme for the real-time management of the considered RAS [63].

by the relevant research community. Furthermore, the Discrete Event Systems (DES) theory [6] provides a powerful base for modeling and analyzing the RAS dynamics. More specifically, Figure 1.1 presents an event-driven control scheme for the considered RAS model. This control scheme responds to the various events taking place in the controlled RAS by updating a state model that defines the feasible behavior of this system. This feasible behavior is "filtered" through the behavioral – or "logical" – controller in order to obtain the admissible behavior, i.e., the behavior that is consistent with certain qualitative specifications imposed on the RAS operation, including the requirements for deadlock-freedom and liveness. Finally, the admissible behavior is processed through the performance-oriented controller in order to select the particular action(s) among the admissible behavior that eventually will be commanded upon the RAS.

As already mentioned, the "logical" control problem of deadlock avoidance that arises in the control scheme of Figure 1.1, is formulated with w.r.t. the "untimed" dynamics of the underlying RAS. Furthermore, this problem has been studied extensively within the DES community, as a particular application of DES Supervisory Control (SC) theory [76,

9]. Furthermore, there has been an additional request for *maximal permissiveness* for the corresponding supervisory control policies; i.e., these policies should ensure the aforementioned capability of all activated processes to run successfully to their completion, while imposing the minimum possible restriction to the original behavior that is generated by the uncontrolled system. Such a maximally permissive supervisor is well-defined and unique for the RAS instantiations studied in [59]. But it is also true that computing the maximally permissive DAP is an NP-hard problem for almost all RAS classes of interest [61].

On the other hand, when it comes to the performance-control problem of the logically controlled RAS, the currently available results are very limited. The research program proposed in this document uses formal DES-based representations of the RAS behavior in order to provide a complete and systematic framework for the RAS scheduling problem. This framework enables a succinct characterization of the involved complexities and their root causes, and the eventual management of these complexities through the development of systematic trade-offs between the operational and computational efficiencies of the control mechanisms and the policies to be deployed on the controlled system.

Formal representational frameworks employed by the current DES theory to represent the basic structure of the workflow dynamics of the considered RAS include the finite state automaton (FSA) [63] and the Petri net (PN) [49, 9] modeling frameworks. These frameworks enable the study of the underlying resource allocation process w.r.t. its deadlock-formation dynamics, and the deployment of real-time control policies that prevent these deadlock formations. And since deadlocks result only from the specific sequencing of the executed resource allocation events and not from the exact timing of these events, the corresponding models and methods must be drawn from the theory of computation [26] that formally models and analyzes "behaviors" represented by such event sequences; i.e., any deadlock-related study of the considered resource allocation processes must focus on their "untimed" dynamics [9] that define a notion of "qualitative behavior" for these processes.

Furthermore, the untimed dynamics and the corresponding deadlock avoidance policies

that were mentioned in the previous paragraph, subsequently must be complemented with timing information and additional control logic that will attend to the aforementioned performance objectives of throughput maximization and/or congestion control. This last task requires an additional set of models that come from the area of stochastic processes [67], and will be able to effectively represent the "timed" dynamics of the underlying resource allocation processes. Even more importantly, there is a need for systematic and effective integration of these new models with the models that support the representation, analysis and control of the untimed behavior. Under further assumption of Markovian behavior for the timing of the various events in the considered RAS, the "timed" dynamics of these systems can be effectively modeled by a generalized stochastic PN (GSPN) [49].

The optimal solution of the developed GSPN models could be calculated through a set of classical methods borrowed from the theory of Markov decision processes [57]. More specifically, a GSPN can be modeled through a Markovian model [2] and its performance can be analyzed through the Markovian model. Therefore, the optimal solution of the considered GSPNs can be obtained through the solution of an average-reward Markov decision process (AR-MDP) whose objective is the maximization of the steady-state average reward. In fact, the underlying logically-controlled RAS can be directly formulated as an AR-MDP [63]. In either way, the develop AR-MDP is used to select one "admissible actions" in the event-driven control scheme of Figure 1.1.

An optimal solution to the considered AR-MDP could be simply calculated through classical approaches such as linear programming, value iteration or policy iteration. However, these classical approaches require the explicit enumeration of the underlying state space, and produces the optimal solution in the form of a look-up table that pairs the RAS decision states with action choices [57]. Clearly, these classical approaches suffer from the "curse of dimensionality" [4], which render their optimal solution intractable. In fact, the size of these state spaces even for moderately sized RAS, when combined with their discrete nature, further implies that even the mere enumeration of an optimal (deterministic)

scheduling policy for the considered AR-MDPs as a look-up table is an intractable task, since this enumeration must specify an optimal action for every single state.

There are some approaches that have tried to address the aforementioned RAS performance-control problem by adapting ideas and techniques that come from the burgeoning area of Approximate Dynamic Programming (ADP) [5, 56]. These approaches reduce the representational and the computational complexity the by approximating some aspects of the underlying MDPs of the considered RASs. In particular, these approaches avoid the explicit enumeration of the state space of the MDPs and they represent the selected solution in a more parsimonious way than a look-up table. But, this complexity reduction is usually caused by the deterioration in performance, or the reduction of the operational efficiency. Hence, there is a trade-off between computational and operational efficiency.

In the context of the considered scheduling problem, [13] has explored the possibility of controlling the very high complexity of this problem by providing a solution to the underlying MDP problem that is based on linear, feature-based approximations of the relative value function. However, such an approach is challenged by (i) (the need for) an *ad hoc* selection of the corresponding features, and (ii) the further fact that it tries to control the quality of the resulting scheduling policy implicitly, by controlling the quality of the approximation of the relative value function. Typically, extensive trial-&-error will be necessary in order to identify a set of features that might strike a satisfactory balance between the quality of the resulting policy and its on-line computational complexity.

Motivated by these remarks, the work of [41, 40] has tried to pursue a more structured approach to the problem, that is based on the ADP notion of "approximation in the policy space" [5]. This new approach employs a timed Petri net (PN)-based representation for the underlying RAS dynamics, and reduces the design of the sought scheduling policy to the problem of determining a set of probability distributions that will govern the selection of an enabled transition at certain markings of the net that correspond to decision points for the underlying resource allocation process. In this new setting, the problem complexity

is controlled by a further "coupling" of the aforementioned distributions that reduces substantially the number of the free variables that are involved in the final problem definition. These free variables define a parameterized policy space, and the selection of an optimized set of values for these variables is effected through stochastic approximation [36]. From the above description, it is clear that the method of [41, 40] enables (a) a much more explicit determination of the employed policy space than the method of [13], and (b) a systematic optimization over this policy space. On the other hand, some apparent limitations of this approach are (i) the eventual formulation of the considered scheduling problem in a more restricted policy space that might not even contain an optimal policy w.r.t. the original MDP formulation, and (ii) the computational challenges that are frequently encountered by stochastic approximation, especially in the case of problems involving a large number of decision variables and the estimation of expectations that are defined over a very large set of possible outcomes.

This research program complements and extends the existing theory on performance control of the considered RAS by introducing a new methodology for developing near-optimal scheduling policies for the considered RAS and the corresponding MDP formulation. From a methodological standpoint, this new approach resolves the performance-control problem that is addressed at each decision epoch by the real-time control framework of Figure 1.1 as follows: First, it defines and solves a linear programming (LP) formulation known as the corresponding "(fluid) LP relaxation", and subsequently it utilizes the obtained optimal solution for this LP in order to define a selection criterion among the set of decisions / actions that are admissible by the applied DAP at the current decision point.

The aforementioned LP relaxation to be solved at each decision epoch is systematically determined by (i) the structure of the underlying RAS, (ii) the RAS state at the current decision epoch, and (iii) the control logic of the applied DAP. The last dependency further implies that the applied DAP must be expressible as a set of linear inequalities on the RAS state; such DAPs are characterized as "linear" in the corresponding literature. The theory

presented in [59] can provide effective and parsimonious realizations of these policies for any given instance from the RAS class(es) to be considered in this work. But one issue that has remained unaddressed by the current literature is the specification and support of a notion of "maximality" within the class of linear DAPs. The systematic investigation of this concept and its properties is another major task of the presented research program.

Furthermore, this research program explores the possibility of further enhancing the FR-based scheduling policy by using techniques from the sensitivity analysis of Markov reward processes and statistical inference. In more specific terms, the developed methodology can be applied at a single decision point of the underlying MDP, in order to detect opportunities for enhancing the performance of the current policy. This detection can be performed through two distinct sequential procedures that are based on the "ranking and selection" algorithm of [32]. The first of these procedures is an "online" scheme that observes the system operation under the current scheduling policy, and it employs the "ranking and selection" algorithm of [32] in order to identify the action with the largest potential at the considered decision point. The second developed procedure is an "offline" procedure that simulates all scheduling policies that result from picking some available action at the considered decision point while maintaining the same decisions for the rest of the policy, and again uses the "ranking and selection" algorithm of [32] in order to identify the policy with the highest long-term average reward.

Although the proposed methodological framework can be applied to any general RAS, the main results of this thesis are developed for the stochastic scheduling problem that concerns the throughput maximization of a particular RAS model that is known as the *capacitated re-entrant line (CRL)*. The CRL model was initially introduced in [64] as a variation of the original re-entrant line (RL) model [34] that assumes finite buffering capacities for all workstations of the line.

The basic RL model has been studied extensively in stochastic scheduling theory; e.g., c.f. [34, 16, 17, 20, 35, 47, 43, 33]. But the works of [64, 12] have shown that the presence

of the finite buffers in the CRL model negates all the past results on the optimal scheduling of conventional re-entrant lines, and introduces all the operational and analytical complications and challenges that were discussed in the earlier parts of this section.

On the other hand, the considered CRL model retains some of the operational and analytical simplicity of a manufacturing flowline, that render it attractive as a prototypical model for the study of the considered scheduling problems. More specifically, an attribute of the (C)RL concept that renders it particularly convenient in that sense, is that it enables a clear and straightforward definition of the notion of the "(long-term) throughput" of the line. At the same time, the considered (C)RL models possess the primary structure that defines the analytical and computational complexity of the corresponding scheduling problems, while avoiding the representational complexity that arises by the presence of more than one process types and more complex resource allocation patterns.

On the more practical standpoint, the uncapacitated "reentrant line" model has been extensively promoted as a pertinent abstraction for the representation of the basic workflow that is encountered in many semiconductor manufacturing fabs [72, 33]. And as the semiconductor manufacturing industry moves toward higher levels of automation, the issue of deadlock avoidance has been revived with the emergence of "cluster tools" [70], and the adopted set of constraints on the buffer capacities that are considered in this work allow the modeling and analysis of the blocking and deadlocking phenomena and their consequences.

In summary, the work presented in this thesis makes the following important contributions to the problem of scheduling the complex resource allocation that takes place in the RAS classes of [59]:

1. First, it defines the notion of "maximal linear" DAP for the considered RAS and investigates its properties.

2. It also provides effective and computationally efficient methodology for obtaining a maximal linear DAP for any given RAS instance.

3. It adapts the generic method of "fluid relaxation (FR)"- based scheduling to the operational context of the logically controlled RAS, focusing primarily on the CRL model.

4. It also uses the modeling framework of timed-continuous PNs [44] in order to further systematize the specification of the necessary FR models for the considered operational setting, and further investigate certain qualitative and quantitative properties of these models.

5. This work also considers the extension of the FR-based scheduling policies developed for the CRL model to some broader RAS classes.

6. Finally, this work also uses results from the sensitivity analysis of Markov reward processes and the ranking and selection algorithm of [32] in order to further improve the scheduling policy obtained through the FR models.

The rest of the thesis is organized as follows: Chapter 2 defines formally the RAS models considered in this work and the corresponding performance optimization problem, and also gives background information that is necessary for building the proposed methodological framework. Chapter 3 addresses the systematic definition of linear DAPs that are appropriate for the considered RAS and observe a "maximality" requirement in terms of their permissiveness. Chapter 4 presents the scheduling methodology for maximizing the throughput of the considered CRL model that is based on the solution of a "fluid" relaxation at each decision point of the original scheduling problem. Chapter 5 enhances the scheduling methodology presented in Chapter 4 by leveraging the modeling and analytical power of timed-continuous Petri nets. This chapter also leverages the increased representational and analytical power that is provided by the timed-continuous Petri net modeling framework in order to identify certain reasons that might compromise the performance of the FR-based scheduling policy. On the other hand, Chapter 6 considers the enhancement of the FR-based scheduling policy through the employment of results coming from the

sensitivity analysis of the underlying Markov reward process and statistical inference. Finally, Chapter 7 concludes the work, and outlines some possible directions for its further extension.

CHAPTER 2
RESEARCH BACKGROUND

This chapter introduces the necessary background information for the development of the methodological framework that was outlined in Chapter 1. First, the modeling abstraction of a disjunctive/conjunctive resource allocation system is formally defined, and some key assumptions that underlie the specification of this model are explicitly stated. Next, there is an overview of the available methodologies for the RAS logical control problem, and the implementation of these methodologies. The third part of this chapter gives the necessary basic knowledge on the considered CRL model, and provides the necessary background material for a detailed characterization of the corresponding logical control problem of deadlock avoidance. Finally, the chapter concludes with the formal definition of the CRL scheduling problem of throughput maximization.

2.1 The considered RAS model

In this section we will focus primarily on the class of Disjunctive-Conjunctive (D/C-) RAS. This is a pretty broad RAS class that allows for (i) an arbitrary structure of the resource requests that are posed by the different processing stages, and also for (ii) the presence of routing flexibility in the supported process plans. A formal definition of the D/C-RAS class is as follows:

Definition 1 *A* Disjunctive-Conjunctive (D/C-) Resource Allocation System (RAS) *is a 4-tuple* $\Phi = \langle \mathcal{R}, C, \mathcal{P}, D \rangle$*, where:*

1. $\mathcal{R} = \{R_1, \ldots, R_m\}$ *is the set of the system* resource types.
2. $C : \mathcal{R} \to \mathbb{Z}^+$ *– the set of strictly positive integers – is the system* capacity *function, characterizing the number of identical units from each resource type available in the*

system. Resources are assumed to be reusable*, i.e., each allocation cycle does not affect their functional status or subsequent availability, and therefore,* $C(R_i) \equiv C_i$ *constitutes a system* invariant *for each* i*.*

3. $\mathcal{P} = \{\Pi_1, \ldots, \Pi_n\}$ *denotes the set of the system* process types *supported by the considered system configuration. Each process type* Π_j *is a composite element itself, in particular,* $\Pi_j =< \Theta_j, \mathcal{G}_j >$*, where: (a)* $\Theta_j = \{\theta_{j1}, \ldots, \theta_{j,l_j}\}$ *denotes the set of* processing stages *involved in the definition of process type* Π_j*, and (b)* $\mathcal{G}_j$ *is an* acyclic digraph *with its node set,* Q_j*, being bijectively related to the set* Θ_j*. Denoting by* $Q_j^{\nearrow}$ *(resp.,* $Q_j^{\searrow}$*) the set of* source *(resp.,* sink*) nodes of* $\mathcal{G}_j$*, the available* process plans *for process type* Π_j *are represented by the* paths *leading from some node* $q_s \in Q_j^{\nearrow}$ *to some node* $q_f \in Q_j^{\searrow}$ *in digraph* $\mathcal{G}_j$*. Also, in the following, we shall set* $\Theta \equiv \bigcup_{j=1}^{n} \Theta_j$ *and* $\xi \equiv |\Theta|$*.*

4. $D : \Theta \rightarrow \prod_{i=1}^{m}\{0, \ldots, C_i\}$ *is the* resource allocation function *associating every processing stage* θ_{jk} *with the* resource allocation vector $D(\theta_{ij})$ *required for its execution; it is further assumed that* $D(\theta_{ij}) \neq \mathbf{0},\ \forall, i, j$*. At any point in time, the system contains a certain number of (possibly zero) instances of each process type that execute one of the corresponding processing stages. A process instance executing a non-terminal stage* $\theta_{ij} \in Q_i \backslash Q_i^{\searrow}$*, must first be allocated the resource differential* $(D(\theta_{i,j+1}) - D(\theta_{ij}))^+$ *in order to advance to (some of) its next stage(s)* $\theta_{i,j+1}$*, and only then will it release the resource units* $|(D(\theta_{i,j+1}) - D(\theta_{ij}))^-|$*, that are not needed anymore. The considered resource allocation protocol further requires that no resource type* $R_i \in \mathcal{R}$ *be over-allocated with respect to its capacity* C_i *at any point in time.*

Finally, for purposes of complexity considerations, we define the size $|\Phi|$ *of RAS* Φ *by* $|\Phi| \equiv |\mathcal{R}| + \xi + \sum_{i=1}^{m} C_i$*.*

Modeling the D/C-RAS dynamics as a Finite State Automaton: The dynamics of

the RAS $\Phi = \langle \mathcal{R}, C, \mathcal{P}, D \rangle$ that was described in the previous paragraph, can be further formalized by a *Deterministic Finite State Automaton (DFSA)* $G(\Phi) = \langle S, E, f, \mathbf{s}_0, S_M \rangle$, that is defined as follows:

1. The *state set* S consists of ξ-dimensional vectors $\mathbf{s}$. The components $\mathbf{s}[l]$, $l = 1, \ldots, \xi$, of $\mathbf{s}$ are in one-to-one correspondence with the RAS processing stages, and they indicate the number of process instances executing the corresponding stage in the considered RAS state. Hence, S consists of all the vectors $\mathbf{s} \in (\mathbb{Z}_0^+)^\xi$ that further satisfy

$$\forall i = 1, \ldots, m, \ \sum_{l=1}^{\xi} \mathbf{s}[l] \cdot D(\theta_l)[i] \leq C_i \tag{2.1}$$

   where, according to the adopted notation, $D(\theta_l)[i]$ denotes the allocation request for resource R_i that is posed by stage θ_l.[1]

2. The *event set* E is the union of the disjoint event sets $E^{\nearrow}$, $\bar{E}$ and $E^{\searrow}$, where:

   (a) $E^{\nearrow} = \{e_{rp} : r = 0, \ \theta_p \in \bigcup_{j=1}^{n} Q_j^{\nearrow}\}$, i.e., event e_{rp} represents the *loading* of a new process instance that starts from stage θ_p.

   (b) $\bar{E} = \{e_{rp} : \exists j \in 1, \ldots, n \text{ s.t. } \theta_p \text{ is a successor of } \theta_r \text{ in graph } \mathcal{G}_j\}$, i.e., e_{rp} represents the *advancement* of a process instance executing stage θ_r to a successor stage θ_p.

   (c) $E^{\searrow} = \{e_{rp} : \theta_r \in \bigcup_{j=1}^{n} Q_j^{\searrow}, \ p = 0\}$, i.e, e_{rp} represents the *unloading* of a finished process instance after executing its last stage θ_r.

3. The *state transition function* $f : S \times E \to S$ is defined by $\mathbf{s}' = f(\mathbf{s}, e_{rp})$, where the

[1] Following standard practice in DES literature (cf., for instance, the relevant definition in page 8 of [9]), in the rest of this document we will frequently use the terms "space" and "subspace" in order to refer to the state set S and its various subsets considered in this work.

components $\mathbf{s}'[l]$ of the resulting state $\mathbf{s}'$ are given by:

$$\mathbf{s}'[l] = \begin{cases} \mathbf{s}[l] - 1 & \text{if } \ l = r \\ \mathbf{s}[l] + 1 & \text{if } \ l = p \\ \mathbf{s}[l] & \text{otherwise} \end{cases}$$

We also notice that $f(\mathbf{s}, e_{rp})$ is a *partial* function, defined only if the resulting state $\mathbf{s}' \in S$. For any state $\mathbf{s} \in S$, the event set $\Gamma(\mathbf{s}) \subseteq E$ for which $f(\mathbf{s}, e)$ is defined, constitutes the set of *feasible events* at $\mathbf{s}$.

4. The *initial state* $\mathbf{s}_0 = \mathbf{0}$, i.e., the state vector with all its components equal to zero. This initial state represents the situation where the system is empty of any process instances.

5. The *set of marked states* S_M is the singleton $\{\mathbf{s}_0\}$. This specification of S_M expresses the requirement for complete process runs.

Letting $\hat{f}$ denote the natural extension of the state transition function f to $S \times E^*$, [2] the behavior of RAS Φ is modeled by the *language* $L(G)$ *generated* by DFSA $G(\Phi)$, i.e., by all strings $\sigma \in E^*$ such that $\hat{f}(\mathbf{s}_0, \sigma)$ is defined. Furthermore, we define the *reachable subspace* S_r of $G(\Phi)$ by

$$S_r \equiv \{\mathbf{s} \in S : \exists \sigma \in L(G) \text{ s.t. } \hat{f}(\mathbf{s}_0, \sigma) = \mathbf{s}\} \tag{2.2}$$

and its *safe subspace* S_s by

$$S_s \equiv \{\mathbf{s} \in S : \exists \sigma \in E^* \text{ s.t. } \hat{f}(\mathbf{s}, \sigma) = \mathbf{s}_0\} \tag{2.3}$$

Also, in the following, we shall denote the complements of S_r and S_s with respect to S by

[2] We remind the reader that, in the relevant automata theory, E^* denotes the *Kleene closure* of the event set E; i.e., E^* contains all the finite-length sequences σ of the elements of E, including the empty sequence ϵ.

$S_{\bar{r}}$ and $S_{\bar{s}}$, and we shall refer to them as the *unreachable* and *unsafe* subspaces. Finally, S_{xy}, $x \in \{r, \bar{r}\},\ y \in \{s, \bar{s}\}$, will denote the intersection of the corresponding sets S_x and S_y.

The target behavior of $G(\Phi)$ and the maximally permissive DAP: The desired – or "target" – behavior of RAS Φ is expressed by the *marked language* $L_m(G)$, which is defined by means of the set of marked states S_M, as follows:

$$\begin{aligned} L_m(G) &\equiv \{\sigma \in L(G) : \hat{f}(\mathbf{s}_0, \sigma) \in S_M\} \\ &= \{\sigma \in L(G) : \hat{f}(\mathbf{s}_0, \sigma) = \mathbf{s}_0\} \end{aligned} \tag{2.4}$$

Equation 2.4, when combined with all the previous definitions, further implies that the set of states that are accessible under $L_m(G)$ is exactly equal to S_{rs}. Hence, we have the following definition of the *maximally permissive deadlock avoidance policy (DAP)* Δ^* for the considered RAS:

Definition 2 *The* maximally permissive deadlock avoidance policy (DAP) Δ^* *for any instantiation Φ from the RAS class of Definition 1 is a supervisory control policy that, at every state $\mathbf{s} \in S_{rs}$, admits a feasible transition $\mathbf{s}' = f(\mathbf{s}, e_{rp})$ of the underlying DFSA $G(\Phi)$ if and only if $\mathbf{s}' \in S_s$.* $\square$

The reader should also notice that the above characterization of the policy Δ^* further implies that, for any given RAS instance Φ, this policy is unique.

2.2 The RAS problem of liveness enforcement and the current literature

According to Definition 2, the maximally permissive DAP Δ^* can be effectively implemented through any mechanism that recognizes and rejects the unsafe states that are accessible through one-step transitions from S_{rs}. In the following, we shall refer to these particular unsafe states as "boundary" unsafe states, and we shall perceive the policy Δ^* as

a classifier that distinguishes effectively between reachable safe states and boundary unsafe states.

The corresponding research community has developed methodology that enables the off-line synthesis of representations for these policies that are very parsimonious, and therefore amenable for real-time control [59]. Among these DAP representations, one of the most interesting and tractable, in terms of, both, analysis and implementation, is that of a "linear" classifier [53, 11]. In this case, the policy admissibility of any given state is resolved based on the ability of this state to satisfy a given set of linear inequalities. In the following, we shall refer to DAPs that admit such a linear representation of their state-acceptance logic as "linear" DAPs.

But as established in [60, 15], linear representation of the maximally permissive DAP is not a viable option for all RAS instantiations of practical interest. To circumvent this limitation, the works of [51, 50, 52, 22] have proposed additional representations for the sought classifiers that either employ nonlinear discriminant functions of the RAS state, or they constitute "non-parametric" classification schemes that rely on the efficient storage and processing of explicit information about the structure of the underlying state space. These alternative representations have been shown to be complete, i.e., they will always provide an effective representation of the target DAP.

Yet, in spite of the aforementioned developments, in many application contexts, DAPs that admit linear representation are still a most desirable solution, due to the analyzability of these policies, and their easy integrability into broader decision-making frameworks. And, in fact, the literature avails of methodology that can synthesize correct linear (but not necessarily maximally permissive) DAPs for a large spectrum of RAS classes of practical interest. Some characteristic examples of this methodology can be found in [21, 65, 37, 55, 42, 71], while a more comprehensive treatment of these methods is provided in Chapter 6 of [59]. But the existing theory does not allow for an explicit characterization and/or control of the extent of the sub-optimality of the DAPs that are derived by it with respect

to the maximally permissive DAP.

A result that has proven very useful in the development of the maximally permissive DAP classifiers, is the following "monotonicity" property that is exhibited by the RAS state safety:

Proposition 1 *Consider the partial order "$\leq$" that is defined on the state space S of any given RAS Φ through the following comparison of the state components:*

$$\forall \mathbf{s}, \mathbf{s}' \in S, \;\; \mathbf{s} \leq \mathbf{s}' \iff (\forall l = 1, \ldots \xi, \;\; \mathbf{s}[l] \leq \mathbf{s}'[l]) \tag{2.5}$$

Then,

1. $\mathbf{s} \in S_s \wedge \mathbf{s}' \leq \mathbf{s} \implies \mathbf{s}' \in S_s$

2. $\mathbf{s} \in S_{\bar{s}} \wedge \mathbf{s} \leq \mathbf{s}' \implies \mathbf{s}' \in S_{\bar{s}}$

□

In [53] it is shown that, thanks to Proposition 1, it is possible to develop a classifier that will distinguish correctly between (a) the states of the reachable and safe subspace S_{rs}, and (b) the boundary unsafe states, by focusing only on the correct classification of the maximal elements of the set S_{rs} and the minimal boundary unsafe states. Furthermore, additional efficiencies in this endeavor, and in the on-line computational complexity of the developed classifier, can be obtained by identifying and removing from the classified vectors any components corresponding to processing stages that do not impact the safety of the system state (e.g., the terminal processing stages of any process type Π_j). The reader is referred to Chapter 4 of [59] for a concise and comprehensive exposition of the corresponding theory on the effective and efficient synthesis of the sought classifiers.

Linear representation of the maximally permissive policy Δ^*: As remarked in the introductory section, a desirable representation of the classification logic that is effected by

the maximally permissive DAP Δ^* is that of a linear classifier. This last concept has been formally defined in [53] as follows:

Definition 3 *Consider two vector sets G and H from a ξ-dimensional vector space V.*

1. *For the need of this work, we shall say that sets G and H are* linearly separated *by a set of k linear inequalities $\{(\mathbf{a}_i, b_i) : i = 1, \cdots, k\}$ if and only if (*iff*)*

$$\begin{aligned}(\forall \mathbf{g} \in G :\ \forall i \in \{1, \cdots, k\},\ \mathbf{a}_i^T \cdot \mathbf{g} \leq b_i) \quad \wedge \\ (\forall \mathbf{h} \in H : \exists i \in \{1, \cdots, k\},\ \mathbf{a}_i^T \cdot \mathbf{h} > b_i)\end{aligned} \tag{2.6}$$

2. *A linear classifier – or separator – for vector sets G and H is* structurally minimal*,* iff *it employs the minimum possible number of linear inequalities that can separate these two sets.*

□

In the case of the classification that is effected by the DAP Δ^*, the roles of the sets G and H in Definition 3 are played, respectively, by the sets $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$ that contain the maximal reachable safe states and the minimal boundary unsafe states. In this case, Proposition 1 implies the following additional result for the sought classifiers [53]:

Proposition 2 *If the maximally permissive DAP Δ^* of a given D/C-RAS Φ admits a representation as a linear classifier of Definition 3, then, there exists such a linear classifier with* nonnegative *parameters $(\mathbf{a}_i, b_i)$ for all the involved inequalities.* □

The astute reader will also notice that Definition 3 implies an asymmetry for the role of the sets $\bar{S}_{rs}$ and $\bar{S}^b_{r\bar{s}}$ in the design of the sought (linear) classifier. This asymmetry is dictated by the further implementation of the resulting classifier through some popular modeling frameworks for the (controlled) RAS dynamics, and especially, the modeling framework of Petri nets (PNs) [49]. In the PN modeling framework, each of the inequalities implementing

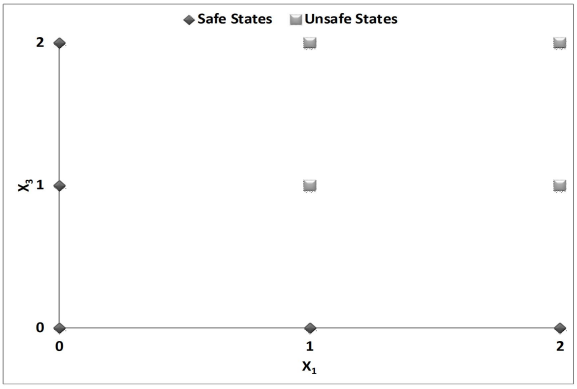

Figure 2.1: Characterization of the safe and unsafe reachable states for an example D/C-RAS with two resource types, R_1 and R_2, with corresponding capacities $C(R_1) = C(R_2) = 2$, and two process types, Π_1 and Π_2, with corresponding process plans $R_1 \rightarrow 2.R_2$ and $R_2 \rightarrow 2.R_1$. Recognizing that the terminal processing stages of these two process types will never get involved in a deadlock, the information that is provided by this figure is projected on the sub-space that is defined by the state components $\mathbf{s}_1$ and $\mathbf{s}_3$, which correspond to the first processing stage of each process plan. Safe reachable states are depicted by rhombi and unsafe reachable states by squares. The reader should notice that the convex hull of the depicted safe states includes the unsafe state corresponding to point $(1, 1)$, and therefore, in this case, the reachable safe states and the boundary unsafe states of the considered system are not linearly separable.

a linear classifier that (a) presents the structure described in Definition 3, and (b) satisfies the additional "non-negativity" condition of Proposition 2, can be enforced on the RAS-modeling PN through the addition of a single place that is known as the corresponding "monitor" place [23, 29]. More importantly, the resulting PN, that represents the behavior of the controlled RAS, belongs to the same class with the PNs that model the uncontrolled RAS behavior, and therefore, it is amenable to the same analysis and design methods that are available for the "plant" (i.e., the RAS-modeling) PN.

On the other hand, it is also well known that Δ^* might not admit a linear representation along the lines of Definition 3 [60, 15].[3] Such a case is provided in Figure 2.1, where it can be seen that the lack of a linear representation for the corresponding DAP Δ^* is due to the inclusion of elements of the set $\bar{S}^b_{r\bar{s}}$ in the convex hull of S_{rs}. As remarked in the earlier part of this section, this problem has been addressed through the development of additional

[3] Also, some interesting related work concerning the limitations of the aforementioned structure of "monitor" places to provide effective representation of the maximally permissive supervisor that ensures deadlock-free and/or live operation for various PN classes, can be found in [28].

representations for the classification logic that is effected by the target policy Δ^*. However, these representations are not amenable to the PN-based implementation of Δ^* that was discussed in the previous paragraph, and to the various analytical and computational possibilities and efficiencies that result from such an implementation. One such possibility that is of particular interest in the presented research program is the inclusion of the logic of the employed DAP into some linear programming formulations that seek to complement the preventive control of deadlock avoidance with scheduling capability, and are known as "fluid relaxations" of the underlying RAS dynamics; these developments are the subject of Chapter 4.

2.3 The CRL and its RAS abstraction

This section introduces the considered CRL model, and provides the necessary background material for a detailed characterization of the corresponding logical control problem of deadlock avoidance.

2.3.1 The considered CRL model

The considered CRL model can be perceived as a subclass of the D/C-RAS of Definition 1 that supports a single process type, with no routing flexibility, and with the only resource types being the servers and the buffer slots of the various workstations of the line.

More specifically, in the considered CRL model, production is supported by L single-server workstations, $W_1, W_2, \ldots, W_L$, each possessing finite buffering capacity B_i, $i = 1, \ldots, L$. On the other hand, the process type that is supported by this line is defined by a sequence of M processing stages, $J_1, J_2, \ldots, J_M$. Each processing stage J_j is carried out at one of the line workstations and it requires a slot of the station buffering capacity during its entire sojourn in it. This station will be denoted by $W(J_j)$, and the function $W(\cdot)$ constitutes the resource allocation function for the corresponding L-SU RAS. Furthermore, it is assumed that $L < M$, an assumption that manifests the re-entrant nature of the line.

Some additional assumptions that detail the line operation, are as follows:

A part visiting the workstation $W(J_j)$ for the processing of the corresponding processing stage J_j will receive service from the station server by having the server visiting the buffer slot that accommodates this part. Hence, any part visiting this workstation will remain in its allocated slot during its entire sojourn at the station, and at any time point during this sojourn, the part will either be waiting for processing, be in processing, or will have completed processing and it will be waiting for transfer to the next required workstation.

Furthermore, in line with the corresponding resource allocation theory, a part that has completed the processing of stage J_j, can move to the next required workstation $W(J_{j+1})$ for the execution of its next processing stage, only when there is an available buffer slot at this workstation. Hence, the processed parts are subjected to blocking effects, and when combined with the re-entrant nature of the considered workflow, these blocking effects can also give rise to deadlocks.[4]

The processing times for the processing stages $J_j,\ j = 1, \ldots, M$, are assumed to be exponentially distributed with mean processing time τ_j. And we shall also set $\mu_j \equiv 1/\tau_j,\ \forall j$.[5] Furthermore, part loading and transfer times between the line workstations are assumed to be negligible w.r.t. processing times.[6]

Finally, since in the following developments our primary objective is the throughput maximization of the considered CRL model, we also assume the existence of an "infinite backlog" of parts waiting for processing in front of the line; i.e., the line never starves for

[4] We also want to notice that while the adopted service model for the line workstations intends to provide a concrete base for the exposition of the presented developments, it is not restrictive in any strong sense, since the presented method can be easily adapted to other service models that are employed by such workstations. From a more practical standpoint, this service model for the line workstations is a quite faithful abstraction of the workflow that is materialized at the various chambers of the, so called, "cluster tools" [70], that constitute a prevailing technology in the current semiconductor manufacturing.

[5] While the assumption of exponential processing times is meant to simplify the exposition of the theory that is developed in this paper, more generally distributed processing times can be handled by approximating them by phase-type distributions to any desired degree of accuracy; please, c.f. to [10] for an introduction to phase-type distributions, and to [9] for a brief introduction on the modeling of non-Markovian dynamics by phase-type distributions.

[6] Non-zero loading and transfer times can be included easily in the considered model through the addition of further stages in the underlying process plan.

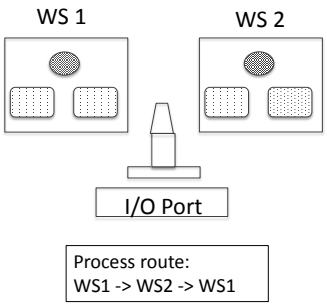

Figure 2.2: An example CRL.

work.

Example: We concretize the definition of the CRL model that was provided in the previous paragraphs, through the example manufacturing system that is depicted in Figure 2.2. This system consists of two workstations, labeled as WS_1 and WS_2 in Figure 2.2, and an I/O port that interfaces it with the rest of its operational environment. Each workstation has a single server, depicted as a grey ellipse in Figure 2.2, and two buffer slots, depicted by the corresponding rectangles. Parts visiting each of the two workstations are accommodated at one of the available buffer slots, and they are processed by the workstation server by having the server visiting the corresponding slot. A robotic manipulator supports the necessary material handling functions, and integrates the entire facility to a fully automated cell. Figure 2.2 also provides the process route for the parts that are processed through this CRL; since workstation WS_1 is visited twice by each part, the considered layout constitutes a re-entrant line. Furthermore, letting $J_j,\ j = 1, 2, 3$, denote the three processing stages of this CRL, under the notation that was introduced in the earlier parts of this section we shall also have $W(J_1) = W(J_3) = WS_1$ and $W(J_2) = WS_2$. Finally, we assume that the processing times for each of the three processing stages are exponentially distributed with corresponding instantaneous rates μ_j and corresponding expected values $\tau_j = 1/\mu_j,\ j = 1, 2, 3$.

2.3.2 Abstracting the CRL "untimed" dynamics through a finite state automaton

Following the relevant theory presented in Section 2.1, we model the basic structure of the workflow dynamics of the introduced CRL model, and the corresponding resource allocation function, by means of a finite state automaton (FSA) $\Phi \equiv (S, E, f, \mathbf{s}_0, S_M)$. Next, we define the various elements of this automaton Φ.

State and state space: A pertinent definition of a notion of *state* for this automaton can be based on the number of the parts waiting for processing, being processed or having completed processing of the different processing stages, J_j, of the supported process type. More specifically:

Definition 4 *The state* $\mathbf{s}$ *of the CRL model considered in this work is a* $3M$*-dimensional vector with component* $3j + k,\ j = 0, \ldots, M-1,\ k = 1, 2, 3$*, denoting respectively the number of parts that are waiting for processing, executing, or having completed processing stage* J_j*.* $\square$

The state set S of the aforementioned automaton Φ consists of all vectors $\mathbf{s}$ that admit an interpretation according to Definition 4, and are compatible with (i) the single-server assumption for the line workstations, and (ii) the available buffering and capacities $B_i,\ i = 1, \ldots, L$, at these workstations. Then, the finite buffering capacity of all workstations implies that the resulting state set S of Φ is, indeed, finite.

Furthermore, since, in the considered CRL model, (a) part loading and unloading require zero time, and (b) there is an infinite backlog of jobs waiting for processing, it is possible to simplify the state concept introduced in Definition 4 by dropping the first and the last component of the state vector $\mathbf{s}$; i.e., parts seeking to execute their first processing stage can be loaded into the line only when the corresponding server is available, and parts having completed processing of the last processing stage can be unloaded immediately. In the following, we shall adopt this simplified state model, with the necessary adjustments in the corresponding notation.

Example: The (simplified) state $\mathbf{s}$ of the FSA Φ corresponding to the CRL of Figure 2.2 is a 7-dim integer vector. The first two components of this vector $\mathbf{s}$, $\mathbf{s}_1$ and $\mathbf{s}_2$, report, respectively, the number of parts at workstation WS_1 executing processing stage J_1 and having completed the processing of this stage; components $\mathbf{s}_3,\ \mathbf{s}_4$ and $\mathbf{s}_5$ report the number of parts in workstation WS_2 that are, respectively, waiting for the execution of stage J_2, executing this stage, and having completed execution of this stage; finally, components $\mathbf{s}_6$ and $\mathbf{s}_7$ of state $\mathbf{s}$ report the number of parts at workstation WS_1 respectively waiting for the execution of stage J_3 and executing this stage. $\square$

Events and their controllability: The set of *events*, E, that advance state $\mathbf{s}$, consists of (i) the event e^l that loads a new part on the line; (ii) the events $e^a_j,\ j = 1, \ldots, M-1$, that advance a part from workstation $W(J_j)$ to the next requested workstation, $W(J_{j+1})$, allocating to this part a free buffer slot of the new workstation; (iii) the events $e^p_j,\ j = 1, \ldots, M$, that initiate the processing of a part at workstation $W(J_j)$ by allocating to it the corresponding server; (iv) the events $e^d_j,\ j = 1, \ldots, M$, that de-allocate the server upon completion of the part processing; and (v) the event e^u that unloads a completed part from the line.

Furthermore, for the needs of the subsequent developments, it is also pertinent to distinguish the various event types that were defined in the previous paragraph into "controllable" and "uncontrollable" events. More specifically, the events of type (i), (ii), (iii) and (v) are *controllable* by the line supervisor. Furthermore, under the aforestated assumptions, these events are executed in zero time when commanded by the supervisor. On the other hand, the events of type (iv) occur spontaneously upon the completion of the processing of the corresponding part, and therefore, they will be treated as *uncontrollable* events. The reader should also notice that, in the *timed* dynamics of the considered CRL, there is a nonzero lag between a type (iii) event and the execution of the corresponding type (iv) event, that corresponds to the necessary processing time.

The state transition function: The state transition function $f : S \times E$ of automaton

Φ is a *partial* function encoding the evolution of the system state $\mathbf{s}$ upon the execution of the different events $e \in E$. In particular, function f is defined only on those pairs $(\mathbf{s}, e) \in S \times E$ where the considered event e is feasible in the corresponding state $\mathbf{s}$. Furthermore, f extends on $S \times E^*$ in the natural manner.

Initial and marked states: For the initial state $\mathbf{s}_0$ of FSA Φ, we set $\mathbf{s}_0 = \mathbf{0}$, i.e., the state where the line is empty of any parts. We also set $S_M = \{\mathbf{s}_0\}$, signifying the fact that an accepting run of FSA Φ should complete all the activated jobs and bring the system back to its initial state.

Deadlock and the need for deadlock avoidance: As remarked in the earlier part of this section, the ability of the considered CRL to reach its marked state $\mathbf{s}_0$ can be compromised by the formation of *deadlock*. In the abstracting representational framework of the above FSA Φ, deadlock is formally defined as follows:

Definition 5 *A CRL* deadlock *is a state* $\mathbf{s}$ *of the corresponding FSA* Φ *where there is a subset* $\mathcal{I} \subseteq \{1, \ldots, L\}$ *such that (i) each workstation* $W_i,\ i \in \mathcal{I}$*, has its buffer slots fully allocated, and (ii) each part* p *accommodated in the workstation subset that is defined by the index set* $\mathcal{I}$ *requests transfer to another workstation in this subset.* $\square$

Example: In the FSA Φ that corresponds to the CRL of Figure 2.2, any state $\mathbf{s}$ that has (a) the buffer slots of workstation W_1 fully allocated to parts executing or having completed their first processing stage, and (b) the buffer slots of workstation W_2 also fully allocated (obviously to parts waiting for the execution / executing / having completed the execution of their second processing stage), is a deadlock. Formally, these deadlock states are represented by the set

$$S^d \equiv \{\mathbf{s} \in S : \mathbf{s}_1 + \mathbf{s}_2 = 2 \ \wedge \ \mathbf{s}_3 + \mathbf{s}_4 + \mathbf{s}_5 = 2\} \tag{2.7}$$

The reader can also check that the states contained in the above set S^d are the only deadlock states of the CRL considered in this example. $\square$

It is clear that under the operational assumptions that were stated in the opening part of this section, parts that are involved in a deadlock formation will be permanently stalled in their current workstations, and at the same time, they will prevent the advancement of any further parts through these workstations. Hence, CRL states containing such deadlock formations must be proactively identified and blocked during the line operation. Next, we overview some basic developments in the RAS deadlock avoidance theory that are particularly relevant to the CRL operational context.

2.3.3 Establishing deadlock freedom for the considered CRL model

An alternative, more compressed representation of the underlying CRL dynamics: It is clear from Definition 5 and its accompanying example that CRL deadlock is due only to the allocation of the workstation buffering capacity, and not to the allocation of the processing capacity of the line servers. Hence, the corresponding problem of deadlock avoidance can be focused on this particular allocation. This can be achieved by considering the further abstraction of the FSA Φ, that was introduced in the previous subsection, to the FSA $\hat{\Phi} = (\hat{S}, \hat{E}, \hat{f}, \hat{\mathbf{s}}_0, \hat{S}_M)$, with a (vector) state $\hat{\mathbf{s}}$ that considers collectively all the parts located at workstation $W(J_j),\ j = 1, \ldots, M$, for the execution of the corresponding processing stage J_j; in other words, each component of the new state $\hat{\mathbf{s}}$ will report the number of parts located at some workstation $W(J_j),\ j = 1, \ldots, M$, for the execution of the corresponding processing stage J_j, without discriminating whether these parts are waiting for processing, are in processing, or have completed processing of this stage and are waiting for transfer to the next required workstation. Furthermore, the event set $\hat{E}$ of $\hat{\Phi}$ will consist only of the type (i), type (ii) and type (v) events of the original FSA Φ. Finally, we also set $\hat{\mathbf{s}}_0 = \mathbf{0}$, and $\hat{S}_M = \{\hat{\mathbf{s}}_0\}$.

Example: For the example CRL of Figure 2.2, the corresponding FSA $\hat{\Phi}$ has a 3-dim state $\hat{\mathbf{s}}$. Furthermore, for any state $\mathbf{s}$ of the original FSA Φ that was defined in Section 2.3.2,

the corresponding state $\hat{\mathbf{s}}$ is obtained through the following equations:

$$\begin{aligned} \hat{\mathbf{s}}_1 &\equiv \mathbf{s}_1 + \mathbf{s}_2 \\ \hat{\mathbf{s}}_2 &\equiv \mathbf{s}_3 + \mathbf{s}_4 + \mathbf{s}_5 \\ \hat{\mathbf{s}}_3 &\equiv \mathbf{s}_6 + \mathbf{s}_7 \end{aligned} \tag{2.8}$$

Clearly, state $\hat{\mathbf{s}}$ changes only when a part enters or leaves one of the line workstations, and it ignores completely the server allocation at these workstations, as well as the specific processing status of the various parts that are located at these workstations.

It is also interesting to notice that, according to Equations 2.7 and 2.8, in the more abstracted representation of the CRL dynamics that is provided by FSA $\hat{\Phi}$, all deadlock formations taking place in the CRL of Figure 2.2 are represented by the single state $\hat{\mathbf{s}}^d = (2, 2, 0)$. It is this representational compression attained by FSA $\hat{\Phi}$ that renders it useful in the analysis of the corresponding deadlock avoidance problem and in the subsequent developments. □

State reachability, safety and maximally permissive deadlock avoidance: In the notational semantics that are associated with FSA $\hat{\Phi}$, we shall further denote by $\hat{S}_r$ the set of *reachable* states of $\hat{\Phi}$, i.e., the states $\hat{\mathbf{s}} \in \hat{S}$ that are accessible from state $\hat{\mathbf{s}}_0$ through some feasible event sequence $\sigma \in \hat{E}^*$. On the other hand, state set $\hat{S}_s$ will denote the set of *co-reachable* – or *"safe"* – states of $\hat{\Phi}$, i.e., the states $\hat{\mathbf{s}} \in \hat{S}$ from which state $\hat{\mathbf{s}}_0$ is accessible through some feasible event sequence $\sigma' \in \hat{E}^*$. We shall also set $\hat{S}_{\bar{r}} \equiv \hat{S} \setminus \hat{S}_r$ and $\hat{S}_{\bar{s}} \equiv \hat{S} \setminus \hat{S}_s$, and we shall refer to these two sets, respectively, as the sets of the *unreachable* and the *unsafe* states. Finally, we shall also use the notation $\hat{S}_{xy} \equiv \hat{S}_x \cap \hat{S}_y$, for $x \in \{r, \bar{r}\}$ and $y \in \{s, \bar{s}\}$.

It should be clear from the definition of the set $\hat{S}_{rs}$ in the previous paragraph that it comprises all the reachable states $\mathbf{s} \in \hat{S}_r$ for which there exist feasible event sequences, $\sigma \in E^*$, leading to the completion of all the parts that are in execution in these states. In

the *state transition diagram (STD)* $\hat{\mathcal{G}}$ representing the dynamics of FSA $\hat{\Phi}$, this property of $\hat{S}_{rs}$ is manifested by the fact that the subgraph induced by its states is the maximal strongly connected component of $\hat{\mathcal{G}}$ containing the empty state $\hat{\mathbf{s}}_0$. These remarks subsequently imply the following characterization of the *maximally permissive DAP* for the considered CRL model:

Theorem 1 *In the representational semantics of FSA $\hat{\Phi}$, deadlock can be avoided while imposing the minimal possible restriction on the workflow dynamics of the underlying CRL, by identifying and blocking attempted transitions from subspace $\hat{S}_{rs}$ to subspace $\hat{S}_{r\bar{s}}$. The resulting DAP is characterized as* maximally permissive *in the corresponding literature, it is uniquely defined, and, in the following, it will be denoted by Δ^*.* $\square$

Theorem 1 is a specialization to the considered CRL model of the corresponding developments of Section 2.2 concerning the characterization of maximally permissive deadlock avoidance in complex resource allocation systems. As we saw in that section, the work of [59] provides also a complete methodology for the effective deployment of the optimal DAP Δ^* for any instantiation of the CRL model that is considered in this work.

Furthermore, the works of [59, 38] present an additional set of results which establish that for a very large subclass – in fact, the majority of the practical instantiations – of the considered CRL model, the optimal DAP Δ^* admits a representation as a set of linear inequalities on the state $\hat{\mathbf{s}}$.

Example: For the example CRL of Figure 2.2, the reader can check that

$$\hat{S}_{\bar{s}} = \hat{S}_d = \{(2, 2, 0)\} \tag{2.9}$$

Hence, for this simple CRL, deadlock can be effectively avoided by enforcing the constraint

$$\hat{\mathbf{s}}_1 + \hat{\mathbf{s}}_2 \leq 3 \tag{2.10}$$

in underlying workflow dynamics. Furthermore, this constraint attains deadlock freedom for the line operation in a maximally permissive manner, since, starting from the initial state $\hat{\mathbf{s}}_0$, the only reachable state that violates this inequality is the deadlock state $\hat{\mathbf{s}}^d = (2, 2, 0)$. $\square$

Correct linear DAPs and policy "lifting" to FSA Φ: As we shall see in Chapter 4, the ability to represent the employed DAP Δ through a set of linear inequalities on the state $\hat{\mathbf{s}}$ is instrumental for developing the LP relaxation and the corresponding scheduling method for the throughput maximization of the considered CRL model that are pursued in this work. In order to address CRL instances where the corresponding maximally permissive DAP does not admit a representation as a set of linear inequalities on state $\hat{\mathbf{s}}$, we also introduce the broader concept of a *"correct linear DAP"*:

Definition 6 *A set of linear inequalities imposed on the state* $\hat{\mathbf{s}}$ *of any given instantiation of the considered CRL model defines a* correct linear *DAP* Δ *for this CRL if and only if the set* $\hat{S}_a(\Delta) \subseteq \hat{S}_r$ *containing the reachable states that satisfy these inequalities, induces a strongly connected component,* $\hat{\mathcal{G}}_a(\Delta)$*, of the corresponding STD* $\hat{\mathcal{G}}$*, that contains the initial state* $\hat{\mathbf{s}}_0$.

Also, the aforementioned state set $\hat{S}_a(\Delta)$ *that is induced by a correct linear DAP* Δ*, is characterized as the (reachable) state (sub-)space that is* admissible *by this policy.*[7] $\square$

The developments of [59] also enable the computation of efficient approximations of the optimal DAP Δ^* that take the form of a correct linear DAP Δ, when the optimal DAP Δ^* does not admit a linear representation.

Furthermore, the eventually employed DAP Δ can be "lifted" to the original FSA Φ, that models more completely the operation of the underlying CRL, through a state admission rule that will admit a state $\mathbf{s} \in S$ if and only if (*iff*) the corresponding state $\hat{\mathbf{s}}$ belongs in $\hat{S}_a(\Delta)$. The resulting admissible subspace of S will be denoted by $S_a(\Delta)$, and the subgraph

[7] This reader should notice that Definition 6 further implies that, for any correct linear DAP Δ, $\hat{S}_a(\Delta) \subseteq \hat{S}_{rs}$.

$\mathcal{G}_a(\Delta)$ induced by the state set $S_a(\Delta)$ in the STD $\mathcal{G}$ of the FSA Φ has similar connectivity properties to the connectivity properties of the subgraph $\hat{\mathcal{G}}_a(\Delta)$ w.r.t. the STD $\hat{\mathcal{G}}$. Furthermore, the notions of "(state) reachability" and "co-reachability / safety" are naturally extended to the CRL dynamics that are described by the FSA Φ.

Table 2.1: The state description for the STD of Figure 2.3.

$\mathbf{s}$	$\mathbf{s}_1\mathbf{s}_2$	$\mathbf{s}_3\mathbf{s}_4\mathbf{s}_5$	$\mathbf{s}_6\mathbf{s}_7$	$\mathbf{s}$	$\mathbf{s}_1\mathbf{s}_2$	$\mathbf{s}_3\mathbf{s}_4\mathbf{s}_5$	$\mathbf{s}_6\mathbf{s}_7$
0	0 0	0 0 0	0 0	33	0 0	0 0 1	1 0
1	1 0	0 0 0	0 0	34	0 0	0 0 0	2 0
2	0 1	0 0 0	0 0	35	1 0	1 0 1	1 0
3	1 1	0 0 0	0 0	36	0 0	0 1 1	1 0
4	0 0	1 0 0	0 0	37	0 0	1 0 0	2 0
5	1 0	1 0 0	0 0	38	0 0	0 1 0	2 0
6	0 0	0 1 0	0 0	39	1 0	0 1 1	1 0
7	1 0	0 1 0	0 0	40	0 1	0 1 1	1 0
8	0 1	0 1 0	0 0	41	1 0	0 0 2	1 0
9	1 0	0 0 1	0 0	42	0 1	0 0 2	1 0
10	1 0	0 0 0	1 0	43	0 1	0 0 2	0 1
11	0 1	0 0 0	1 0	44	0 1	0 0 2	0 0
12	0 0	1 0 0	1 0	45	0 1	0 1 1	0 1
13	0 1	0 0 0	0 1	46	0 1	0 1 1	0 0
14	0 0	1 0 0	0 1	47	0 0	1 1 0	1 0
15	0 0	0 1 0	0 1	48	0 1	0 1 0	0 1
16	0 0	0 0 1	0 1	49	0 0	1 1 0	0 1
17	0 0	0 0 0	1 1	50	0 0	1 1 0	0 0
18	0 0	0 0 0	1 0	51	1 0	1 1 0	0 0
19	0 0	0 0 0	0 1	52	0 1	1 1 0	0 0
20	1 0	1 0 0	1 0	53	1 0	1 0 1	0 0
21	0 0	0 1 0	1 0	54	1 0	0 1 1	0 0
22	1 0	0 1 0	1 0	55	0 1	1 0 1	0 0
23	0 1	0 1 0	1 0	56	0 1	1 0 0	1 0
24	1 0	0 0 1	1 0	57	0 0	2 0 0	1 0
25	0 1	0 0 1	1 0	58	0 1	1 0 0	0 1
26	0 0	1 0 1	1 0	59	0 0	2 0 0	0 1
27	0 1	0 0 1	0 1	60	1 0	2 0 0	1 0
28	0 0	1 0 1	0 1	61	1 0	1 1 0	1 0
29	0 0	0 1 1	0 1	62	0 1	1 1 0	1 0
30	0 0	1 0 0	1 1	63	0 1	1 1 0	0 1
31	0 0	0 1 0	1 1	64	0 1	1 0 1	0 1
32	0 0	0 0 1	1 1	65	1 1	0 1 0	0 0

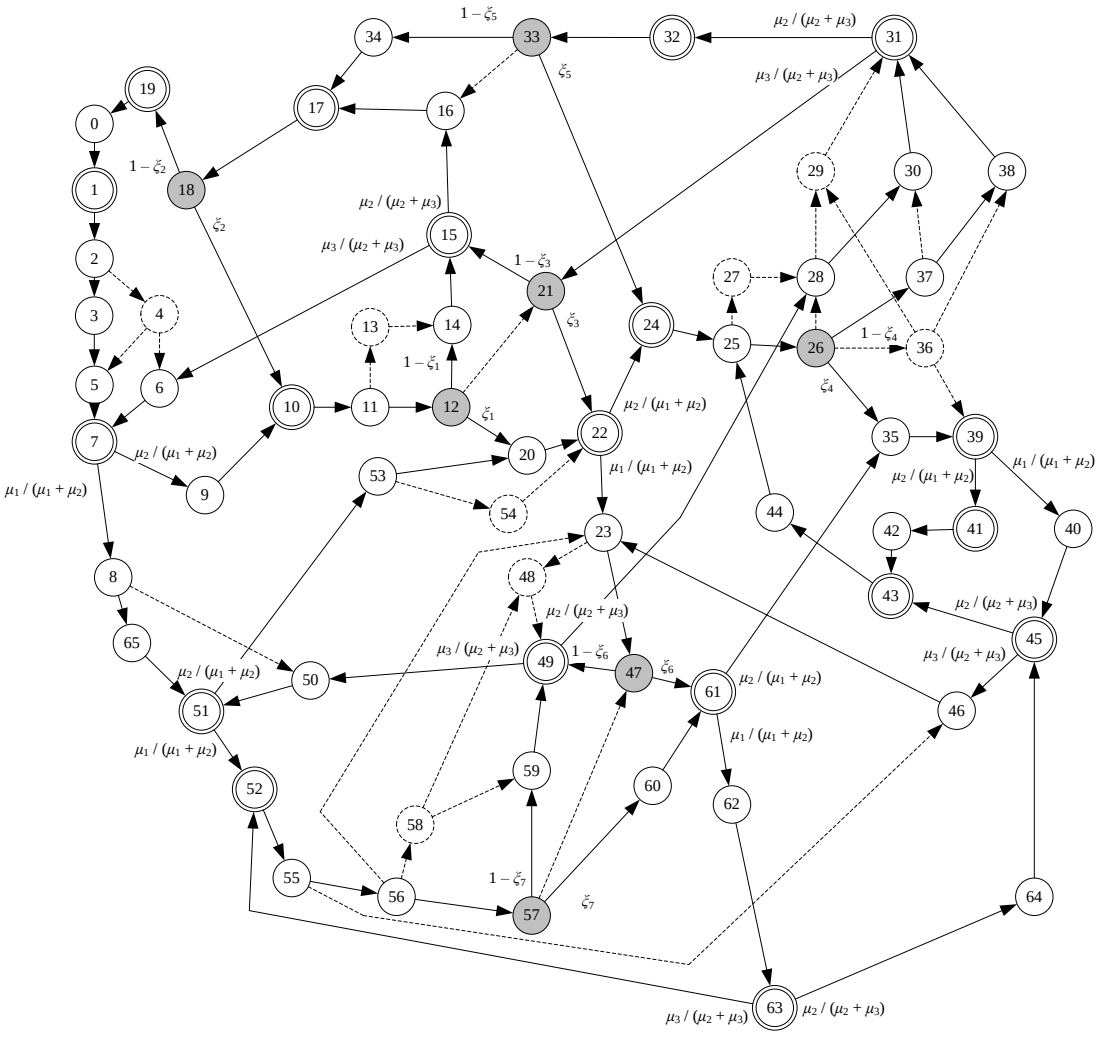

Figure 2.3: The reachable and safe state space S_{rs} for the CRL of Figure 2.2, and some further structure that defines the MDP characterizing the corresponding throughput-maximization problem.

Example: Figure 2.3 depicts the reachable and safe state space, S_{rs}, for the example CRL of Figure 2.2. S_{rs} is also the subspace admitted by the maximally permissive, correct, linear DAP, Δ^*, that is defined by Equation 2.10. A complete characterization of the various states depicted in this figure is provided in Table 2.1. In the next section, we shall show how to formulate the scheduling problem of the throughput maximization for this CRL as an MDP, by introducing additional information to the STD of Figure 2.3 that pertains to the "timed" dynamics of the considered CRL.

2.4 The CRL scheduling problem of throughput maximization and its MDP formulation

Introducing "timed" dynamics to FSA Φ – tangible and vanishing states: This section (i) introduces the scheduling problem of the throughput-maximization of the considered CRLs when operated under the supervision of a correct DAP Δ, and (ii) formulates this problem as an average-reward MDP. To fully characterize this scheduling problem, and proceed with the corresponding MDP formulation, we must augment the FSA-based representation of the workflow dynamics of the considered CRLs with time-related elements. An effective way to perform this augmentation is by differentiating the states $\mathbf{s}$ in the Δ-admissible state space S_a [8] into (a) states where the only enabled events are some uncontrollable events e_j^d, and (b) states that enable controllable events as well,[9] according to the following definition:

Definition 7 *Consider a CRL instance controlled by a correct DAP Δ. Then, a state $\mathbf{s} \in S_a$ is characterized as* tangible *iff the only enabled events in $\mathbf{s}$ are some uncontrollable events e_j^d; otherwise, state $\mathbf{s}$ will be characterized as* vanishing. *Furthermore, the entire set of tangible states will be denoted by S_a^T, and the set of vanishing states will be denoted by S_a^V.* $\square$

We demonstrate the concepts of "tangible" and "vanishing" states that were introduced in the previous definition, and highlight the significance of these concepts for the subsequent developments, through the following example.

Example: In the example STD of Figure 2.3, tangible states are depicted as double-circled, while vanishing states are single-circled.

Next, let us focus on tangible state #7. From Table 2.1, it is clear that this state contains two parts: a part p_1 executing processing stage J_1, and a part p_2 executing processing stage

[8] In order to avoid an "over-loading" of the employed notation, in the following we shall use S_a instead of $S_a(\Delta)$, assuming that this set is defined by an appropriately selected DAP Δ.

[9] The application of the employed DAP Δ ensures that every state $\mathbf{s} \in S_a$ will possess at least one enabled event that is also admissible by the applied DAP Δ.

J_2. The earlier completion of each of these two parts w.r.t. the other one leads respectively to states #8 and #9. Furthermore, since processing times for these two parts are exponentially distributed with respective instantaneous rates μ_1 and μ_2, the respective probabilities for each of these two transitions are those annotated in the figure. Finally, the expected sojourn time for state #7 is $1/(\mu_1 + \mu_2) > 0$.

On the other hand, states #8 and #9 are vanishing states, since in each case, the completed part can be advanced to its next processing stage, obtaining, respectively, states #65 and # 10. Furthermore, under the stated operational assumptions for the considered CRL, these part advancements require zero time; hence, the sojourn time for states #8 and #9 is zero. □

As revealed in the previous example, tangible states essentially define an "exponential race" among its enabled events e_j^d, and therefore, they possess a non-zero sojourn time. On the other hand, since (i) vanishing states enable some controllable event, according to Definition 7, and (ii) any controllable event in the considered CRL model executes in zero time, the sojourn times of these states will be consistently equal to zero.[10]

Furthermore, it is important to notice that, while in the case of tangible states the selection of the executed events is resolved endogenously, by the corresponding exponential race, in the case of vanishing states, there is a further need for an extraneous mechanism that will select a particular enabled controllable event for execution. This mechanism must bias the underlying selection in a way that it supports the pursued objective of throughput maximization, and it will be provided by the sought scheduling policy. Next we discuss how this scheduling policy can be obtained, at least in principle, through the formulation and solution of a *Continuous-Time, Average-Reward (CT-AR) MDP* [57] that is defined by means of the various structural elements that have been introduced in this section.

Formulating the considered scheduling problem as an MDP: According to the general MDP theory [57], in order to obtain a complete MDP formulation of the considered

[10]These remarks also justify the respective names of these two state classes as "tangible" and "vanishing"; the corresponding terminology has been borrowed from [1].

scheduling problem in the context of the timed CRL dynamics that were presented in the previous part of this subsection, we need to define: (i) the decision states of this MDP; (ii) the set of actions that are available at each decision state; (iii) the transitional dynamics that are incurred by the execution of a particular action at a decision state; (iv) the immediate rewards that result from these executions; and (v) the function of these rewards that formalizes the problem objective. Next, we provide a detailed characterization of all these elements; our discussion relies heavily on the various concepts and insights that were provided in the earlier parts of this chapter.

When it comes to the *"decision states"* of the considered MDP, it should be evident from the discussion that was provided in the previous part of this section, that these states are the vanishing states that result from the occurrence of an event e_j^d at any of the admissible tangible states $\mathbf{s} \in S_a^T$. The following definition formalizes this remark.

Definition 8 *Let $X \subseteq S_a$ denote the set of states that result from the execution of an event e_j^d at some state $\mathbf{s} \in S_a^T$. Then, the set X constitutes the set of* the decision states *of the considered MDP.* $\square$

Next, we define the set of the available *"actions"* (or *"decisions"*) at any given state $\mathbf{s}' \in X$. From a more conceptual standpoint, each of these decisions will take the considered MDP to a new admissible tangible state $\mathbf{s}'' \in S_a^T$, in zero time, and then, the process will wait for the next completion event, e_j^d, at that state. It is also important to notice that in order to reach a next admissible tangible state $\mathbf{s}'' \in S_a^T$ from the current decision state $\mathbf{s}'$, the underlying process might have to pass through a cascade of vanishing states that are reached through a sequence of controllable events, σ; the reader is referred to Figure 2.3 and the accompanying Table 2.1 for some concrete examples of this last statement. Furthermore, the above remarks motivate the following definition:

Definition 9 *For any decision state $\mathbf{s}' \in X$, define the corresponding* "tangible reach" *of $\mathbf{s}'$, $\mathcal{TR}(\mathbf{s}')$, as the set of the admissible tangible states $\mathbf{s}''$ that are reachable from state $\mathbf{s}'$*

through an event sequence $\sigma \in E^*$ that contains only controllable events. Then, the set of actions*, $\mathcal{A}(\mathbf{s}')$, of the considered MDP at decision state $\mathbf{s}'$ is defined as $\mathcal{A}(\mathbf{s}') \equiv \mathcal{TR}(\mathbf{s}')$.* $\square$

For any state $\mathbf{s}'' \in \mathcal{TR}(\mathbf{s}')$, the corresponding action can be materialized through the execution of any controllable-event sequence σ leading from state $\mathbf{s}'$ to state $\mathbf{s}''$.

We also notice, for completeness, that the set of vanishing states $\mathbf{s}'''$ which are reachable from state $\mathbf{s}'$ through the aforementioned event sequences σ that lead to some state $\mathbf{s}'' \in \mathcal{TR}(\mathbf{s})$, is characterized as the *"vanishing reach"* of state $\mathbf{s}'$. This set of states is denoted by $\mathcal{VR}(\mathbf{s}')$, and it can be empty for some states $\mathbf{s}'$.

The *"transitional dynamics"* for the considered CT-MDP model that result from the execution of an action $a \equiv \mathbf{s}'' \in \mathcal{TR}(\mathbf{s}')$ at some decision state $\mathbf{s}'$, are determined by the exponential race that takes place in the tangible state $\mathbf{s}''$.

On the other hand, since our stated objective is the maximization of the long-term throughput of the line, the *expected immediate reward*, $r(\mathbf{s}', a)$, from executing action a at state $\mathbf{s}'$ is defined as follows:

Definition 10 *For the considered MDP, the* expected immediate reward *from executing action a at state $\mathbf{s}'$ is denoted by $r(\mathbf{s}', a)$, and it is equal to the probability that the next decision state will be defined by the occurrence of event e^d_M that corresponds to the completion of the last processing stage by a running part and the unloading of this part from the line. Hence, letting $\mathbf{s}'' \in \mathcal{TR}(\mathbf{s}')$ denote the tangible state that corresponds to action a, and $\mathcal{E}(\mathbf{s}'')$ denote the set of the events e^d_j enabled in $\mathbf{s}''$, the expected immediate reward $r(\mathbf{s}', a)$ will be equal to $\mu_M / \sum_{e^d_j \in \mathcal{E}(\mathbf{s}'')} \mu_j$ if $e^d_M \in \mathcal{E}(\mathbf{s}'')$, and zero otherwise.*

Finally, in view of the above definitions of the decision states and actions of the considered MDP, the induced transitional dynamics, and the expected immediate rewards, the problem of maximizing the throughput of the considered CRLs is reduced to the problem

of maximizing the (long-term) average reward of this MDP.[11] The communicating structure of the admissible state space S_a for the underlying stochastic process that was established in the earlier parts of this section, further implies that this CT-MDP formulation is well defined, and it will have an optimal solution that takes the form of a deterministic, stationary policy [57]. Hence, letting Π denote the set of deterministic stationary policies for the considered MDP, an optimal schedule for the considered CRL is represented by a policy $\pi^* \in \Pi$ that, at each state $\mathbf{s}' \in X$, will select a single action $a \in \mathcal{TR}(\mathbf{s}')$ so that

$$\pi^* = \arg\max_{\pi \in \Pi} \lim_{N \to \infty} \frac{1}{E[t_N]} E\left[\sum_{i=1}^{N} r(\mathbf{s}'_i, a_i) \;\middle|\; \mathbf{s}_0, \pi \right]$$

In the above equation, t_N denotes the time of the N-th state transition of the underlying stochastic process. Furthermore, some simplification of this CT-MDP formulation, and some methodology for its solution through uniformization [9], are presented in Appendix A of [39]. But, as remarked in the introductory chapter, in most practical cases the solution of this CT-MDP model will be intractable due to the very large size of the involved state spaces. Hence, there is a remaining need for the computation of suboptimal scheduling policies that will trade off some of the performance of the underlying system for computational tractability. The development of such a methodology is at the core of the presented research program. We conclude this section by discussing the MDP formulation and its optimal solution for the example CRL of Figure 2.2.

Example: As already discussed in the previous parts of this section, the STD of Figure 2.3 highlights the classification of the depicted states into tangible and vanishing, and

[11] The specification of the set of actions at each decision state $\mathbf{s}'$ of this CT-MDP through the corresponding tangible reach $\mathcal{TR}(\mathbf{s}')$ implies a *non-idling* scheduling policy for the underlying CRL; i.e., under such a policy, no server that could be engaged in the processing of some available part will remain idle. Due to the blocking experienced in the operation of the considered CRLs, such a non-idling scheme might be suboptimal [24, 64]. We have opted to confine the presented developments within the class of the non-idling scheduling policies, in an effort to attain some simplicity for the presentation of the main concepts and ideas involved. But it is possible to extend the presented methodology to *deliberately idling* schemes, by introducing further actions at the states $\mathbf{s}'$ that correspond to decision epochs; these actions will correspond to controllable-event sequences σ leading to some state $\mathbf{s}''$ in the vanishing reach $\mathcal{VR}(\mathbf{s}')$ of the considered state $\mathbf{s}'$, that contains some enabled events e_j^d.

it also reports the transitional dynamics that are defined by the exponential races that take place at each tangible state. An additional development in Figure 2.3 is a proposed "thinning" of the presented STD through the elimination of the vanishing states and their interconnected transitions that are depicted in dashed lines. This simplification is justified by the facts that (i) the timed performance of the considered CRL is determined by the sojourn times that are spent by this line at the tangible states only, and (ii) the proposed "thinning" does not alter the reachability among the various tangible states, and also among the decision states $\mathbf{s} \in X$ that result from the execution of an e_j^d–type event.[12]

The vanishing states that involve choice in the remaining STD structure are the seven states colored in grey in Figure 2.3. More specifically, in the remaining STD, each of these seven states possesses two enabled transitions, and any selection between these two transitions can be represented by setting the corresponding variable $\xi_k,\ k = 1, \ldots, 7$, either to the value of 0 or 1. In the context of the corresponding MDP terminology, any possible pricing of the variables ξ_k depicted in Figure 2.3 defines a complete deterministic stationary policy for the considered MDP.[13]

On the other hand, to fully specify this MDP, we must also specify the immediate-reward function. Under the previously introduced notation, this function is fully defined by associating with every pair $(\mathbf{s}', \mathbf{s}'') \in X \times \mathcal{TR}(\mathbf{s}')$, an immediate reward equal to the occurrence probability of event $e_3^d (\equiv e^u)$ in state $\mathbf{s}''$.

Finally, for a better understanding of the semantics of the MDP that was defined in the previous paragraphs, we also notice that, in the operational context of the CRL depicted in

[12] A complete analysis that formalizes this "thinning" process, provides a more rigorous justification for it, and supports it with computationally efficient algorithms, is presented in [40].

[13] In more accurate terms, the suggested pricing of the variables ξ_k defines a deterministic stationary policy of the considered MDP because of the following two additional facts: (I) For any vanishing state $\mathbf{s}' \in X$, that results from the execution of an e_j^d–type event at some tangible state $\mathbf{s}$, the aforementioned pricing of the variables ξ_k defines completely the next tangible state $\mathbf{s}''$ to be reached from state $\mathbf{s}'$. (II) In addition, for any pair of states $\mathbf{s}'_1$ and $\mathbf{s}'_2$ in X, the pricing of the variables ξ_k does not introduce any "coupling" in the resulting transitional dynamics from these two states.

We also notice that allowing the variables ξ_k to take values in the interval $[0, 1]$ would result in a randomized stationary scheduling policy. But in the considered problem setting, enabling such a randomization will not lead to any performance enhancement for the underlying CRL [57, 12].

Figure 2.2, each of the variables ξ_k that define the various deterministic stationary policies for this MDP, essentially models the choice between (i) allocating the server of workstation WS_1 to a part that will execute processing stage J_3 (by setting $\xi_k = 0$), and (ii) allocating this server to a newly loaded part for the execution of its first processing stage (by setting $\xi_k = 1$).

CHAPTER 3

MAXIMAL LINEAR DEADLOCK AVOIDANCE POLICIES FOR D/C-RAS

In this chapter, first we introduce the new DAP class of the maximal linear DAPs, explaining the rationale that underlies the definition of these policies, and establishing their well-posedness for any given instance from the considered RAS classes. Next, we address the more practical issue of computing maximal linear DAPs for any given RAS instance. Finally, we complement the theoretical developments by (i) some further discussion on an efficient implementation of the proposed algorithms, and (ii) a series of numerical experiments that exemplify these developments, and demonstrate and assess their computational tractability.

3.1 Maximal linear DAPs

This section introduces the new concept of the "maximal linear DAP", as it materializes in the considered class of D/C-RAS. We shall formally define this new DAP class by providing a complete set of conditions that must be satisfied by the admissible subspace of such a policy. Hence, let Δ denote a tentative DAP from the considered class for some given D/C-RAS Φ, and let $S_a(\Delta) \subseteq S$ denote the corresponding policy-admissible subspace. We also define $S_{\bar{a}}(\Delta) \equiv S \setminus S_a(\Delta)$. For the dynamics of the controlled system to be well-defined, clearly we need

$$\mathbf{s}_0 \in S_a(\Delta) \tag{3.1}$$

Then, we can also define $S_r(\Delta)$, the reachable subspace of Φ under policy Δ, as the limit set of the following recursion:

$$S_r(\Delta)^{(0)} := \{\mathbf{s}_0\} \tag{3.2}$$

$$S_r(\Delta)^{(k+1)} := S_r(\Delta)^{(k)} \cup \{\mathbf{s}' \in S_a(\Delta) : \exists \mathbf{s} \in S_r(\Delta)^{(k)}, e \in \Gamma(\mathbf{s}) \text{ with } f(\mathbf{s}, e) = \mathbf{s}'\} \tag{3.3}$$

A primary requirement in the specification of the sought policy Δ is that it does not induce any new deadlocks or livelocks; such a DAP is characterized as "correct" in the relevant literature [59]. The correctness of Δ translates into the following requirement for the corresponding set $S_r(\Delta)$:

$$\forall \mathbf{s} \in S_r(\Delta),\ \exists e \in \Gamma(\mathbf{s}) \setminus E^{\nearrow},\ f(\mathbf{s}, e) \in S_r(\Delta) \tag{3.4}$$

In more natural terms, the condition of Equation 3.4 requires that at every state $\mathbf{s}$ that is reachable in the considered RAS under supervision by Δ, there is a policy-admissible event e that concerns the stage advancement or the unloading of an already initiated process instance. In Chapter 6 of [59] it is shown that this condition further implies that the subgraph $\mathcal{G}_r(\Delta)$ of the state transition diagram (STD) of the FSA $G(\Phi)$ that is induced by the state set $S_r(\Delta)$, contains the initial state $\mathbf{s}_0$ and it is strongly connected. Hence, state $\mathbf{s}_0$ is reachable from every state $\mathbf{s} \in S_r(\Delta)$, and therefore, there will be no deadlocks or livelocks in the operation of the controlled RAS.

Next we address the requirement that the sought policy Δ will admit a representation through the linear classifiers of Definition 3. Furthermore, for the reasons that were explained in Section 2.2, we also want the linear representations for our target policies Δ to satisfy the "non-negativity" property of Proposition 2.

To formally state the conditions that will help us meet these two requirements, let us denote the convex hull of any given vector set V by $conv(V)$, and also define the set $S_{\bar{r}}^{b}(\Delta)$

as follows:

$$S_{\bar{r}}^{b}(\Delta) \equiv \{\mathbf{s}' \in S_r \setminus S_a(\Delta) : \\ \exists \mathbf{s} \in S_r(\Delta),\ e \in \Gamma(\mathbf{s}) \text{ with } f(\mathbf{s}, e) = \mathbf{s}'\} \tag{3.5}$$

The set $S_{\bar{r}}^{b}(\Delta)$ contains all the states $\mathbf{s}$ that are reachable through a single transition from $S_r(\Delta)$ but are blocked by policy Δ. Hence, this set collects all the "boundary inadmissible" states in the controlled dynamics of RAS Φ.

Then, in analogy to the corresponding results for the maximally permissive DAP Δ^*, the aforestated requirement for a representation of the policy Δ through a linear classifier of Definition 3 with non-negative coefficients can be met by introducing the following two conditions to the policy specification:

$$\forall \mathbf{s}, \mathbf{s}' \in S_a(\Delta),\ \mathbf{s}' \leq \mathbf{s}\ \wedge \mathbf{s} \in S_a(\Delta) \Longrightarrow \mathbf{s}' \in S_a(\Delta) \tag{3.6}$$

$$conv(S_r(\Delta)) \cap S_{\bar{r}}^{b}(\Delta) = \emptyset \tag{3.7}$$

Up to this point, we have articulated the requirements that must be satisfied by the sought DAP Δ for any given D/C-RAS Φ so that (i) it is correct, and (ii) admits a desired linear representation, as qualified by Definition 3 and the condition of Proposition 2. Policy Δ will also be a *"maximal"* (correct) linear DAP for RAS Φ, if there is no other correct linear DAP Δ' for RAS Φ with an admissible reachable subspace $S_r(\Delta')$ such that $S_r(\Delta') \supset S_r(\Delta)$.

The following definition provides a more formal expression to all the previous discussion.

Definition 11 *A policy Δ is a* linear *DAP for some given D/C-RAS Φ* iff *its admissible subspace $S_a(\Delta)$ satisfies the following conditions:*

Correctness: $(\mathbf{s}_0 \in S_a(\Delta)) \wedge$

$$\big(\forall \mathbf{s} \in S_r(\Delta),\ \exists e \in \Gamma(\mathbf{s}) \setminus E^{\nearrow},\ f(\mathbf{s}, e) \in S_r(\Delta)\big)$$

Monotonicity: $\forall \mathbf{s}, \mathbf{s}' \in S_a(\Delta),$

$$\mathbf{s}' \leq \mathbf{s} \ \wedge \mathbf{s} \in S_a(\Delta) \Longrightarrow \mathbf{s}' \in S_a(\Delta)$$

Linearity: $conv(S_r(\Delta)) \cap S_{\bar{r}}^b(\Delta) = \emptyset$

Furthermore, a linear DAP Δ for a given D/C-RAS Φ is maximal iff *there is no other linear DAP Δ' for D/C-RAS Φ with $S_r(\Delta') \supset S_r(\Delta)$.* $\square$

Example: Two maximal linear DAPs for the example D/C-RAS of Figure 2.1, are the DAPs Δ^1 and Δ^2 that will admit a state $\mathbf{s} \in S$ if its projection on the 2-dim space that is defined by the state components $\mathbf{s}_1$ and $\mathbf{s}_3$, belongs, respectively, in the sets $S_a^1 \equiv \{(0,0),(1,0),(2,0),(0,1)\}$ and $S_a^2 \equiv \{(0,0),(1,0),(0,1),(0,2)\}$.

Indeed, both of these policies admit the initial state $\mathbf{s}_0$ and it can be easily checked that they do not suffer from any policy-induced deadlock or livelock. Furthermore, they satisfy the "monotonicity" requirement of Definition 11, and the corresponding state sets $S_r(\Delta^i), S_{\bar{r}}^b(\Delta^i),\ i = 1, 2$, will admit linear separation in the projected space that is defined by the state coordinates $\mathbf{s}_1$ and $\mathbf{s}_3$. Finally, these two policies are also maximal, since the only possible expansion of the corresponding sets $S_r(\Delta^i),\ i = 1, 2$, is by re-admitting the blocked pairs (0,2) and (2,0) in the corresponding sets $S_a^i,\ i = 1, 2$; but the policy that will result from any of these two augmentations is Δ^*, and we know that this policy is not linear.

Finally, the reader should also notice that $S_r(\Delta^1) \neq S_r(\Delta^2)$, and therefore, the two policies Δ^1 and Δ^2 are essentially different.

Existence but non-uniqueness of maximal linear DAPs: The closing remark in the previous example further implies that, for any given D/C-RAS Φ, the maximal linear DAPs of Definition 11 will not be unique, in general. Hence, for further reference, we shall denote the set of linear DAPs for any given D/C-RAS Φ by $\mathcal{L}(\Phi)$, and its subset that contains its maximal elements by $\bar{\mathcal{L}}(\Phi)$.

The next result is also important for the well-posedness of the considered DAP class.

Proposition 3 *For any given D/C-RAS Φ, $\bar{\mathcal{L}}(\Phi) \neq \emptyset$.*

Proof: For any given D/C-RAS Φ, consider the policy $\hat{\Delta}$ that admits a state $\mathbf{s} \in S$ *iff* (a) either it is the initial state $\mathbf{s}_0$, or (b) it contains only one active process instance. Then, it is easy to see that the policy $\hat{\Delta}$ is correct, and satisfies the "monotonicity" requirement of Definition 11. It is also clear that the admissibility logic of this policy can be expressed by the linear inequality

$$\sum_{i=1}^{\xi} \mathbf{s}[i] \leq 1$$

Hence, the set of linear DAPs for any given D/C-RAS Φ, $\mathcal{L}(\Phi)$, is non-empty. Since this set is also finite, it will possess well-defined maximal elements, and therefore, the set $\bar{\mathcal{L}}(\Phi)$ is also non-empty. $\square$

With the notion of the maximal linear DAP well-defined, next we turn to the development of the necessary algorithms that will provide a maximal linear DAP for any given D/C-RAS Φ.

3.2 Computing the maximal linear DAPs

In this section, we consider the computation of the maximal linear DAPs, for any given D/C-RAS Φ, that were defined in the previous section. The presented developments are organized as follows: (i) First, we introduce a basic algorithm for the systematic enumeration of all the maximal linear DAPs $\Delta \in \bar{\mathcal{L}}(\Phi)$, for any given D/C-RAS Φ, and we prove the algorithm correctness and the finiteness of its computation. (ii) In a second part of this

Algorithm 1 The main algortihm for computing $\bar{\mathcal{L}}(\Phi)$

Input: DFSA $G(\Phi)$
Output: $\bar{\mathcal{L}}(\Phi)$

/* INITIALIZE */
$STORE$:= NILL; $EXPLORE := \langle \bar{S}_{rs} \rangle$;
/* MAIN ITERATION */
while $EXPLORE \neq NILL$ **do**
  $\bar{S}_r :=$ POP($EXPLORE$);
  $S^b_{\bar{r}} := \{\mathbf{s} \in S : \big(\exists \mathbf{s}' \in S,\ \mathbf{s}'' \in \bar{S}_r,\ e \in \Gamma(\mathbf{s}') \text{ s.t. } f(s', e) = \mathbf{s} \text{ AND } \mathbf{s}' \leq \mathbf{s}''\big)$ AND $\big(\nexists\, \mathbf{s}''' \in \bar{S}_r \text{ s.t. } \mathbf{s} \leq \mathbf{s}'''\big)\}$;
  $\bar{S}^b_{\bar{r}} := \{\mathbf{s} \in S^b_{\bar{r}} : \nexists \mathbf{s}' \in S^b_{\bar{r}} \text{ s.t. } \mathbf{s}' \neq \mathbf{s} \text{ AND } \mathbf{s}' \leq \mathbf{s}\}$;
  if (($\bar{S}_r, \bar{S}^b_{\bar{r}}$) linearly separable) AND ($\nexists\, \bar{S}'_r \in STORE : \bar{S}'_r \supseteq \bar{S}_r$) **then**
    Remove from $STORE$ any element sets $\bar{S}''_r$ s.t. $\bar{S}''_r \subset \bar{S}_r$;
    Enter $\bar{S}_r$ in $STORE$;
  else
    for all $\mathbf{s} \in \bar{S}_r$ **do**
      $\tilde{S}_r := PRUNE(\bar{S}_r,\ \mathbf{s},\ G(\Phi))$;
      if ($\nexists\, \bar{S}'_r \in STORE :\ \bar{S}'_r \supseteq \tilde{S}_r$) **then**
        PUSH($\tilde{S}_r; EXPLORE$);
      end if
    end for
  end if
end while
/* TERMINATE */
return $STORE$;

section, we also discuss certain implementational details that concern the employed data structures and some additional arbitrating logic that can be introduced at certain steps of the algorithm, and can streamline further the overall execution. (iii) Finally, the last part of the section provides the results of some numerical experimentation that seeks to assess (a) the practical tractability of the presented algorithm, and (b) the extent of restrictiveness that is introduced by the linearity of the target policies.

3.2.1 A basic algorithm for the enumeration of the set $\bar{\mathcal{L}}(\Phi)$

Description of the proposed algorithm: The basic structure for the algorithm that we propose for the enumeration of the policy set $\bar{\mathcal{L}}(\Phi)$, for any given D/C-RAS Φ, is presented

Algorithm 2 Function $PRUNE(\bar{S},\ \tilde{\mathbf{s}},\ G(\Phi))$

Input: DFSA $G(\Phi)$, maximal-state set $\bar{S}$, pruned state $\tilde{\mathbf{s}}$
Output: $PRUNE(S, G(\Phi))$

$\hat{S}_r := \{\mathbf{s}_0\}$;
while $\hat{S} := \{\mathbf{s} \in S \setminus (\hat{S}_r \cup \{\tilde{\mathbf{s}}\}) : (\exists \mathbf{s}' \in \hat{S}_r,\ e \in \Gamma(\mathbf{s}') \text{ with } f(\mathbf{s}', e) = \mathbf{s}) \text{ AND } (\exists \mathbf{s}'' \in \bar{S} \text{ s.t. } \mathbf{s} \leq \mathbf{s}'')\} \neq \emptyset$ **do**
    $\hat{S}_r := \hat{S}_r \cup \hat{S}$;
end while
while $\hat{S} := \{\mathbf{s} \in \hat{S}_r :\ \forall e \in \Gamma(s) \setminus E^{\nearrow},\ f(\mathbf{s}, e) \notin \hat{S}_r\} \neq \emptyset$ **do**
    $\hat{S}_r := \hat{S}_r \setminus \hat{S}$;
end while
$\bar{S}_r := \{\mathbf{s} \in \hat{S}_r : \nexists \mathbf{s}' \text{ s.t. } \mathbf{s}' > \mathbf{s}\}$;
return $\bar{S}_r$;

in the pseudo-code of Algorithm 1. This algorithm starts with the computation of the set of reachable and safe states, S_{rs}, that defines the reachable subspace under the maximally permissive DAP Δ^*, and seeks to detect all the maximal subsets of this set – including the set S_{rs} itself – that will define correct linear DAPs. In particular, the set S_{rs} is tested first, and if it is found to admit a linear representation, then the algorithm exits returning this set as the single element of the set $\bar{\mathcal{L}}(\Phi)$. On the other hand, if the maximally permissive DAP Δ^* is not linearly representable, the algorithm will run a search process for all those proper subsets of S_{rs} that constitute the reachable subspace for a maximal linear DAP $\Delta \in \bar{\mathcal{L}}(\Phi)$.

Each proper subset of S_{rs} that is considered by this search process, is obtained from a "parent" subset in the generated "search tree" by removing (i) a single maximal element of the "parent" set, and (ii) any additional states that need to be removed in order to restore the correctness of the induced DAP. The induced DAP Δ that is obtained through this processing can be tested for membership in $\mathcal{L}(\Phi)$ through the corresponding algorithms that are available in [53]. More specifically, Algorithm 1 computes the sets $\bar{S}_r$ and $\bar{S}_r^b$, containing, respectively, the maximal reachable states and the minimal boundary inadmissible states under the considered policy Δ, and tries to construct a linear separator for these two sets using one of the methods that are presented in [53]. If such a construction is successful, then the algorithm recognizes policy Δ as a linear DAP; otherwise, policy Δ defines a

"branching node" of the underlying search tree for the generation of new candidate policies Δ' according to the state-removal scheme that was mentioned at the beginning of this paragraph. The detailed branching logic that was employed in our eventual implementation of Algorithm 1 is discussed in the next subsection.

Another salient point for the complete understanding of the pseudo-code that is presented in Algorithm 1, is that the aforementioned subsets of S_{rs} that are generated during the search process, are supposed to be represented by means of their maximal elements; in the presented pseudo-code, this fact is indicated by "barring" or "tilding" the corresponding sets. Some details on the buildup and the maintenance of these particular representations during the algorithm execution are provided in the next subsection.

The "mechanics" of the search process that was described in the previous paragraphs, are facilitated in Algorithm 1 through the employment of two lists, $STORE$ and $EXPLORE$, that hold, respectively, (a) the subsets of S_{rs} that correspond to linear DAPs and are maximal among the currently detected such sets, and (b) subsets of S_{rs} that have been generated as potential candidates for specifying maximal linear DAPs, but have not been assessed and further processed yet. Then, as can be seen in Lines 3–16 of Algorithm 1, the detailed processing of a set $\bar{S}_r$ that has been extracted from the list $EXPLORE$, consists of the following steps: First it is checked whether this set defines a linear DAP.[1] If this is the case, and, furthermore, this set is not dominated by any set already in $STORE$, then it is entered in $STORE$ as the reachable subspace of a tentative maximal linear DAP. During this stage, $STORE$ is also cleared by any already stored sets that are dominated by the new entrance. If, on the other hand, the considered set does not specify a linear DAP, then it spawns a number of entries for the list $EXPLORE$. Each of these entries is generated through (i) the removal of a maximal element from the "parent" set, and (ii) the further pruning of the resulting set in order to ensure that it specifies a correct DAP. The function that performs this pruning is listed in Algorithm 2, and it constitutes a "fixed point"

[1] As already mentioned, this test can be performed through the procedures that have been developed in [53].

computation that seeks to establish the correctness condition of Definition 11.

The entire algorithm is initialized with list $STORE$ empty and list $EXPLORE$ containing the set S_{rs} (represented by the subset of its maximal elements, $\bar{S}_{rs}$). Hence, the algorithm will first assess whether the maximally permissive DAP Δ^* is a linear DAP, and if this is the case, it will terminate without considering any other policies. In the opposite case, it will run as described in the previous paragraphs, and eventually it will terminate when the list $EXPLORE$ becomes empty. At this point, the algorithm will return the contents of the $STORE$ list as its output.

Concluding the description of Algorithm 1, we also notice, for completeness, that the set dominance that is tested in certain parts of the algorithm, can be resolved by means of the maximal elements that are stored in the employed representation of these sets, through the following criterion:

$$\bar{S}_r \supseteq \bar{S}'_r \iff \forall \mathbf{s}' \in \bar{S}'_r,\ \exists \mathbf{s} \in \bar{S}_r :\ \mathbf{s} \geq \mathbf{s}' \tag{3.8}$$

Proving the correctness of Algorithm 1 and the finiteness of its computation: Next we proceed to prove the correctness of Algorithm 1 and the finiteness of its computation. In order to derive these results, we shall start with a more technical proposition that will ensure that the policies Δ induced by the sets $\bar{S}_r$ that are generated and stored in the lists $EXPLORE$ and $STORE$ of Algorithm 1, satisfy the "monotonicity" requirement of Definition 11. In order to establish this result, we must also specify more explicitly the sets $S_a(\Delta)$ that consist of all the admissible states by any such policy Δ. For the needs of the subsequent discussion, we shall define $S_a(\Delta)$, for any policy Δ that is induced by the search process of Algorithm 1, as follows:

$$S_a(\Delta) \equiv \left\{\mathbf{s} \in S_r : \exists \mathbf{s}' \in \bar{S}_r \text{ s.t. } \mathbf{s} \leq \mathbf{s}'\right\} \cup S_{\bar{r}s} \tag{3.9}$$

The first set of states in the right-hand-side of Equation 3.9 contains all the reachable states of the FSA $G(\Phi)$ that are dominated by some element of the policy-defining set $\bar{S}_r$, and therefore, are admissible by the corresponding policy Δ according to the logic of Algorithm 1 that was discussed in the previous part of this subsection. The second set is the set of all the safe but unreachable states of $G(\Phi)$. Since these states are unreachable in the original dynamics of the FSA $G(\Phi)$, they will never materialize when this FSA is controlled under the considered policy Δ. But their inclusion in the set $S_a(\Delta)$ is technically necessary in order to ensure that this set will contain the entire sub-lattices of $(\mathbb{Z}_0^+)^\xi$ that are dominated by each of its elements (since some elements of these sub-lattices might be unreachable). With the sets $S_a(\Delta)$ well-defined through Equation 3.9, now we can state and prove the following result:

Proposition 4 *The sets $S_a(\Delta)$ corresponding to the policies Δ that are generated by Algorithm 1 through Equation 3.9, satisfy the "monotonicity" condition of Definition 11.*

Proof: We shall establish the result of Proposition 4 through a double induction, where the outer induction will run on the sets that enter list $EXPLORE$, and the inner induction will be with respect to the state sets that are pruned during the iterations that take place in Algorithm 2. Also, in the rest of the proof, we shall write S_a instead $S_a(\Delta)$, in order to simplify the corresponding notation.

As the base case for the outer part of the pursued induction, we notice that the set S_a that is induced by the first state set to enter list $EXPLORE$, during the initialization phase of Algorithm 1, is the set $S_{rs} \cup S_{\bar{r}s} = S_s$, which satisfies the "monotonicity" condition of Definition 11 by Proposition 1.

Next, suppose that the "monotonicity" property is possessed by the first k entries to list $EXPLORE$, and consider the set S_a that corresponds to entry $(k+1)$ to this list. We shall show that this set possesses the desired "monotonicity" property by showing that, for every state $\mathbf{s}$ pruned from this set by function $PRUNE$, $PRUNE$ will also prune all states $\mathbf{s}''$ with $\mathbf{s}'' > \mathbf{s}$.

We establish this result, using the second induction. For the base case of this induction, we establish the aforementioned result for those states that are removed by the first execution of the "While" loop of Lines 5–7 in $PRUNE$. Hence, let $\mathbf{s}'$ denote the maximal state that was removed from the "parent" set S'_a during the generation of the considered set S_a, and $\mathbf{s}$ denote a state that was removed from the newly generated set S_a during the first iteration of the "while" loop in Algorithm 2.

From the logic that drives this pruning process, it follows that the set $\Gamma(\mathbf{s}) \cap (E \backslash E^{\nearrow})$ is a singleton, containing an event e that leads to the (removed maximal) state $\mathbf{s}'$. Furthermore, event e cannot be an unloading event, since, then, we shall have $\mathbf{s} > \mathbf{s}'$, and $\mathbf{s}'$ is not maximal. Since event e is not a loading event either, it follows that the pruned state $\mathbf{s}$ must contain the same set of process instances with state $\mathbf{s}'$.

Next, we use the above facts to show that any state that dominates state $\mathbf{s}$ in S'_a will also be pruned from S_a by the pruning function of Algorithm 2. Hence, consider a state $\mathbf{s}'' \in S'_a$ with $\mathbf{s}'' > \mathbf{s}$. According to the D/C-RAS dynamics, the presence of the extra processes in state $\mathbf{s}''$ can only hinder the further progress of its common processes with state $\mathbf{s}$. Hence, for this last set of processes, the only available move in $\mathbf{s}''$ is the event e that led from $\mathbf{s}$ to $\mathbf{s}'$. If this event can be executed in $\mathbf{s}''$ while some of the extra processes are still present in the system, then, for the resulting state $\mathbf{s}''''$, it will hold $\mathbf{s}''' > \mathbf{s}'$, which contradicts the presumed maximality of $\mathbf{s}'$ in S'_a. Hence, the extra processes in state $\mathbf{s}''$ must be cleared before the remaining processes in that state can move. But then, any process-completing path for state $\mathbf{s}''$ must still go through state $\mathbf{s}$, and therefore, state $\mathbf{s}''$ must be pruned as well.

For the inductive step of the inner induction, suppose that the inductive hypothesis holds for the first k executions of the "while" loop of function $PRUNE$, and consider a state $\mathbf{s}$ that is pruned during the $(k+1)$ iteration of this loop. If this state is maximal in S'_a, then, its removal from S_a cannot impair the "monotonic" structure of this set. If, on the other hand, there exists $\mathbf{s}' \in S'_a$ s.t. $\mathbf{s}' > \mathbf{s}$, then, we discern two cases for the advancement of the extra processes in $\mathbf{s}'$: If the advancement of these processes is interleaved with the

advancement of some processes that are also present in state $\mathbf{s}$, then, any such interleaving will result in a state $\mathbf{s}''$ that dominates the corresponding state $\mathbf{s}'''$ that would result from the advancement of the same processes in $\mathbf{s}$. But state $\mathbf{s}'''$ is a state that must have been removed in the previous iterations of the considered "while" loop in the $PRUNE$ function, and, according to the inductive hypothesis, state $\mathbf{s}''$ has been pruned as well. If, on the other hand, the extra processes in state $\mathbf{s}'$ must complete before any further advancement of the common processes in $\mathbf{s}'$ and $\mathbf{s}$, then, the completion of these extra processes would result in state $\mathbf{s}$, which is pruned according to the working hypothesis. Hence, state $\mathbf{s}'$ must also be pruned. □

Next, we state and prove the main technical result of this subsection, that concerns the correctness and the finiteness of the computation of Algorithm 1.

Theorem 2 *When applied on any given D/C-RAS Φ, Algorithm 1 will terminate in a finite number of steps, and it will return a nonempty output that is a correct enumeration (under the adopted representation) of the set $\bar{\mathcal{L}}(\Phi)$.*

Proof: The finiteness of the algorithm computation results from the following facts: At each iteration, the generated sets $\bar{S}_r$ that enter list $EXPLORE$ for further processing are of finite number and of smaller cardinality than the corresponding "parent" set that generated them. Also, the starting set $\bar{S}_{rs}$ is a finite set, and each generated subset of this set will be processed through the $EXPLORE$ list a finite number of times. Finally, each single operation that is performed by the algorithm is also of finite length.

Next, we prove the correctness of the algorithm, i.e., that the algorithm will compute correctly the target set $\bar{\mathcal{L}}(\Phi)$. We have already seen that the first set $\bar{S}_r$ that is considered by Algorithm 1 as the reachable subspace for a candidate policy Δ, is the set $\bar{S}_{rs}$, that induces the maximally permissive DAP, Δ^*, according to Definition 2. Hence, the algorithm will return the maximally permissive DAP Δ^* as its unique output, if this policy is also found to be linear.

On the other hand, if Δ^* does not admit a linear representation, and the algorithm must search for alternative policies, then, Line 11 of the algorithm, together with the definition of function $PRUNE$ in Algorithm 2, ensure that every set $\tilde{S}_r$ that enters the lists $EXPLORE$ and (possibly) $STORE$, induces a policy Δ that satisfies the "correctness" condition of Definition 11. Also, Proposition 4 ensures that all these policies satisfy the "monotonicity" condition of Definition 11. Similarly, the first condition in the "if" statement in Line 6 of the algorithm ensures that a policy Δ will enter list $STORE$ only if it is linear. Finally the way that the sets $\bar{S}_r$ are generated by pruning their "parent" sets by one maximal element at a time, together with the set-inclusion tests that are performed in Lines 6 and 7 of the algorithm, ensure that the eventual content of list $STORE$ will be the maximal subsets of the set S_{rs} that pass the aforementioned tests.

In order to complete the "correctness" part of the proof, we must also establish that there is no advantage in removing a non-maximal element from the "parent" sets that are handled by the pursued search process over the subsets of S_{rs}. This can be seen upon noticing that these state removals essentially seek to separate the convex hulls of the resulting sets $\bar{S}_r$ and $\bar{S}^b_{\bar{r}}$ for the corresponding induced policy Δ. But it is clear that any state removal that maintains intact the "parent" set $\bar{S}_r$ will not be able to effect the aforementioned separation.

Finally, the claimed non-emptiness of the $STORE$ list that is returned by Algorithm 1, follows from the algorithm correctness, that was established in the previous paragraphs, and Proposition 3, that established the existence of (maximal) linear DAPs for every D/C-RAS Φ.

3.2.2 Further implementational details

From a computational standpoint, the policy enumeration that is effected by Algorithm 1 is a very expensive proposition. Hence, in this subsection, we provide some further discussion on certain implementational details that can streamline the algorithm and expedite its execution on sizable problem instances. This discussion focuses primarily on (i) the

Table 3.1: The main data structures that are employed by class `RAS` for the representation of the underlying RAS dynamics and the various policies Δ evaluated by the considered algorithm.

Class `RAS`: Static Part
`ReachableStates`: A dynamic list containing all reachable states of the considered RAS. `NextStates`: A list of hashsets mapping each reachable state to the set of states that are immediately reachable from it. `PrevStates`: A list of hashsets mapping each reachable state to the set of reachable states that are immediately backward reachable from it. `ReachableStatesSet`: A hashset that enables testing of membership in list `ReachableStates`, for any state $\mathbf{s}$, in $O(1)$. `DominatedBy`: A hashmap mapping each reachable state to a hashset of the reachable states that dominate it by one extra process instance; this data structure is used to facilitate the computation of maximal safe states and minimal unsafe states.
Class `RAS`: Dynamic Part (provides effective representation to some tentative policy Δ)
`IsAdmissible`: A list of Boolean entries used to mark each state as admissible or not by the considered policy Δ. This is set S_r in the semantics of Algorithm 1. `MaxAdmissible`: A hashset of the maximal admissible states under the considered policy Δ. This is set $\bar{S}_r$ in the semantics of Algorithm 1. `MinBoundaryInadmissible`: A hashset of the minimal boundary inadmissible states under the considered policy Δ. This is set $\bar{S}_r^b$ in the semantics of Algorithm 1. `InseparableMinBoundaryInadmissible`: A hashset of the minimal boundary inadmissible states that are not linearly separable from the maximal admissible states under the considered policy Δ.

employed data structures for the representation of the dynamics of the underlying RAS and of the policies that are evaluated at the different stages of the conducted search process, (ii) the "branching logic" that drives this search process through the selection of the maximal states to be pruned from the admissibility space of each policy Δ that fails to pass the linearity test, and (iii) the management of the $EXPLORE$ list that provides a sense of direction in this search process. We address each of these issues in a separate part.

The main data structures used for the implementation of the algorithm and their maintenance: In our implementation of Algorithm 1 we organized all the information processed by the algorithm in a class named `RAS` that consisted of two parts:

- A "static" part that is shared by all the instances of this class, and provides an effective and efficient representation of the state transition diagram of the underlying RAS. This part is computed only once at the beginning of the algorithm execution.
- A "dynamic" part that is distinct for each object that instantiates this class, and provides an effective and efficient representation of the tentative policies Δ that are generated by the conducted search process.

Both parts of class `RAS` are detailed in Table 3.1. In the algorithm semantics that were introduced in Subsection 3.2.1, each instance of class `RAS` essentially collects all the information that is stored and processed at a single node of the search tree that is generated by Algorithm 1. Next, we briefly discuss how the different data structures that appear in Table 3.1 are built and maintained by the algorithm every time that it generates an instance from class `RAS`.

The first four data structures in the static part of class `RAS` are constructed through standard reachability analysis on the state space of the underlying RAS Φ during the initial phase of the algorithm execution. At this point, the hashmap `DominatedBy` is also easily constructed by a procedure that scans the list `ReachableStates`, and for each state $\mathbf{s}$ that is stored in it, generates the states $\mathbf{s}_i = \mathbf{s} + \mathbf{e}_i,\ i = 1, \ldots, \xi$, and checks (through the hashset `ReachableStateSet`) whether state $\mathbf{s}_i$ is a reachable state.

On the other hand, the list `IsAdmissible` belonging to the first node of the search tree that is built by Algorithm 1, is obtained through standard co-reachability analysis on the STD of the underlying RAS Φ with respect to the initial state $\mathbf{s}_0$. This computation will return the set S_{rs} as the admissible set of states for the first tentative policy Δ to be considered by the algorithm. Furthermore, the obtained list `IsAdmissible` is processed

Algorithm 3 Calculates Maximal Admissible States

Input: `IsAdmissible`
Output: `MaxAdmissible`

```
 1: for all s in IsAdmissible do
 2:    TEST := TRUE;
 3:    for all s′ in DominatedBy[s] do
 4:       if IsAdmissible[s′] then
 5:          TEST := FALSE; GOTO step 8;
 6:       end if
 7:    end for
 8:    if TEST then
 9:       MaxAdmissible.add(s);
10:    end if
11: end for
12: return MaxAdmissible;
```

through Algorithm 3 in order to obtain the hashset `MaxAdmissible` that collects its maximal elements. The reader should notice that by making use of the pre-constructed hashmap `DominatedBy`, Algorithm 3 computes the sought set `MaxAdmissible` with complexity $O(N \cdot \xi)$, where N is the cardinality of its input set and ξ is the dimensionality of the stored state vectors; a more traditional approach based on a pairwise comparison of the admissible states would be of order $O(N^2 \cdot \xi)$.

For the first node of the search tree of Algorithm 1, the hashet `MinBoundaryInadmissible` is obtained by feeding the set of inadmissible states to Algorithm 4; this last set can be obtained straightforwardly from the content of list `IsAdmissible`. The first part (Lines 1–12) of this algorithm identifies the set of the boundary inadmissible states by checking their one-step reachability from some admissible state, while the second part (Lines 13–19) uses the `DominatedBy` hashmap in order to identify the minimal elements of this last set.

Once the `MinBoundaryInadmissible` hashset has been constructed, the algorithm proceeds to assess whether the considered policy Δ – in the case of the first node of the underlying search tree, this is the maximally permissive policy Δ^* – admits a linear representation. This test is performed by checking whether each identified minimal bound-

Algorithm 4 Calculates Minimal Boundary Inadmissible States

Input: STATES
Output: `MinBoundaryInadmissible`

```
 1: MinBoundaryInadmissible := STATES;
 2: for all s in STATES do
 3:    PrevAdmissible := FALSE;
 4:    for all s' in PrevStates[s] do
 5:       if IsAdmissible[s'] then
 6:          PrevAdmissible := TRUE;
 7:       end if
 8:    end for
 9:    if ¬ PrevAdmissible then
10:       MinBoundaryInadmissible.remove(s);
11:    end if
12: end for
13: for all s in MinBoundaryInadmissible do
14:    for all s' in DominatedBy[s] do
15:       if MinBoundaryInadmissible.contains(s') then
16:          MinBoundaryInadmissible.remove(s');
17:       end if
18:    end for
19: end for
20: return MinBoundaryInadmissible;
```

ary inadmissible state $\mathbf{u}$ is linearly separable from the maximally admissible states $\mathbf{s}_i \in \bar{S}_r$. More specifically, for any given minimal boundary inadmissible state $\mathbf{u}$, this last test is performed through the solution of the following LP:

$$\max_{\mathbf{a}\geq\mathbf{0},b\geq 0} 0 \tag{3.10}$$

s.t.

$$\mathbf{a}^T\mathbf{s}_i \leq b\ ,\ \forall \mathbf{s}_i \in \bar{S}_r \tag{3.11}$$

$$\mathbf{a}^T\mathbf{u} \geq b+\epsilon \tag{3.12}$$

In the above formulation, ϵ is a preselected parameter such that $\epsilon \to 0^+$. The resulting LP formulation is essentially a feasibility test for the constraints that appear in it. A negative

outcome for this test implies that the considered minimal boundary inadmissible state $\mathbf{u}$ is not linearly separable from the set of the maximal admissible states, and therefore, state $\mathbf{u}$ is placed in the `InseparableMinBoundaryInadmissible` hashset. Similarly, the non-emptiness of this last set at the end of this entire computation implies that the considered policy Δ is not linearly separable, and therefore, the corresponding node in the search tree will be a branching node. Otherwise, the considered policy Δ belongs in $\mathcal{L}(\Phi)$, and the corresponding instance of class `RAS` that represents it, enters the $STORE$ list of Algorithm 1.

In order to complete the first part of this subsection, it remains to discuss how we have organized the computation of the dynamic part of the class `RAS` for the internal nodes of the underlying search tree, in order to take advantage of (i) the "locality" of the pruning process that is effected by Algorithm 2, and (ii) the information that is available in the instances of the class `RAS` that represents their "parent" nodes.

This computation starts with the execution of Algorithm 5. Algorithm 5 (i) first constructs the `RAS` instance of the newly generated node (to be called the "child" node in the following) by replicating the `RAS` instance of its parent node, (ii) performs the pruning of the state $\mathbf{s}$ that generates this new node by eliminating state $\mathbf{s}$ from the `IsAdmissible` list of this node, and subsequently (iii) it proceeds to perform the remaining state pruning that is dictated by Algorithm 2. This further pruning utilizes the information on state admissibility that is provided in the `IsAdmissible` list of the parent node[2] and the structure of the underlying automaton Φ that is encoded in the static part of the `RAS` data structure. In particular, for any already pruned state $\mathbf{u}$, Algorithm 5 identifies additional states $\mathbf{x}$ that need to be pruned, by looking into the `PrevStates` hashset of $\mathbf{u}$ for members of this list that do not have any (remaining) admissible states in their `NextStates` hashset. Any such state $\mathbf{x}$ is removed from the `IsAdmissible` list, it is entered into list Q for further processing according to the logic that was described above, and it is also entered into list

[2]This information has been inherited by the child node through the aforementioned replication of the corresponding `RAS` object.

Algorithm 5 Constructs a Child Node from its Parent Node and Updates State Admissibility for the Child Node

Input: parent `RAS` instance, pruned state $\mathbf{s}$
Output: child `RAS` instance

```
 1: child := parent
 2: child.IsAdmissible[s] := FALSE;
 3: Q := {s}; E := {s};
 4: while Q ≠ NILL do
 5:    u := Q.pop();
 6:    for all x in PrevStates[u] do
 7:       TEST := FALSE;
 8:       for all y in NextStates[x] do
 9:          if child.IsAdmissible[y] then
10:             TEST := TRUE;
11:          end if
12:       end for
13:       if ¬ (TEST ∨ E.contains(x)) then
14:          child.IsAdmissible[x] := FALSE;
15:          Q.add(x); E.add(x);
16:       end if
17:    end for
18: end while
19: return child;
```

E in order to be recognized as an already identified inadmissible state.

Once the child `IsAdmissible` list has been properly updated, it is run through Algorithm 3 that will compute its maximal elements, i.e., the `MaxAdmissible` hashset of the child node. On the other hand, in order to compute the `MinBoundaryInadmissible` hashset of this node, the algorithm first combines (i) the `MinBoundaryInadmissible` hashet of the parent node, and (ii) the set of states that were pruned during the execution of Algorithm 5, into a new hashset that is called `PotentialMinBoundaryInadmissible`, and it is this set that is fed into Algorithm 4. Finally, the assessment of the linearity of the policy Δ that is induced by the derived `IsAdmissible` list, and the computation of the corresponding `InseparableMinBoundaryInadmissible` hashset, are performed as described in the previous parts of this subsection.

Concluding the first part of this subsection, it should be clear from all the above dis-

cussion that the data structures that are defined in Table 3.1, manage, indeed, to support a localized and incremental computation of all those elements that define each policy Δ considered by the proposed algorithm, and also enable a srtaightforward assessment of the linearity of these policies.

Refining the "branching" logic of Algorithm 1: In the original statement of Algorithm 1, the algorithm is supposed to branch on all the elements of the set $\bar{S}_r$, i.e., on each maximal admissible state $\mathbf{s}_i$ by the policy Δ that corresponds to the current search node (c.f. Line 10 of Algorithm 1).

However, in our eventual implementation of Algorithm 1 we have confined the nodal branching only to a certain subset of the original set $\bar{S}_r$, focusing on those elements of $\bar{S}_r$ that can substantially improve the prospects of obtaining a linear policy Δ' from the corresponding state pruning. The rationale for the selection of the target subset of $\bar{S}_r$ that is eventually used for the proposed nodal branching, is defined by the following two remarks.

Remark 1: Let $\bar{S}'_r = \{\mathbf{s} \in \bar{S}_r : \mathbf{s} \text{ is an extreme point of } conv(S_r)\}$. Then, the sets $(\bar{S}_r, \bar{S}^b_{\bar{r}})$ are linearly separable if and only if the sets $(\bar{S}'_r, \bar{S}^b_{\bar{r}})$ are linearly separable.

Indeed, if the sets $(\bar{S}_r, \bar{S}^b_{\bar{r}})$ are linearly separable, then the sets $(\bar{S}'_r, \bar{S}^b_{\bar{r}})$ are linearly separable since $\bar{S}'_r$ is a subset of $\bar{S}_r$. On the other hand, if the sets $(\bar{S}_r, \bar{S}^b_{\bar{r}})$ are not linearly separable, then the sets $(\bar{S}'_r, \bar{S}^b_{\bar{r}})$ are also not linearly separable, because it is easy to see that the set $conv(S_r)$ is essentially determined by the set $\bar{S}'_r$.

In order to calculate the set $\bar{S}'_r$, we iterate over all states $\mathbf{s_i} \in \bar{S}_r$, and for each of these states we solve the following linear program that checks whether this state can be expressed as a convex combination of the other states in $\bar{S}_r$:

$$\min_{\mathbf{x} \geq \mathbf{0}} x_i \tag{3.13}$$

s.t.

$$\sum_{k:\mathbf{s}_k \in \bar{S}_r} x_k \mathbf{s}_k \geq \mathbf{s}_i \tag{3.14}$$

$$\sum_{k:\mathbf{s}_k \in \bar{S}_r} x_k = 1 \tag{3.15}$$

If the optimal solution of the above LP is $x_i = 1$, we can conclude that the state $\mathbf{s}_i$ is an extreme point of $conv(S_r)$, and we add the state $\mathbf{s}_i$ to the set $\bar{S}'_r$. The constructed set $\bar{S}'_r$ can be further thinned through the following remark.

Remark 2: Let the set $\bar{S}''_r$ be the subset of $\bar{S}'_r$ that is obtained by Algorithm 6. Then, the sets $(\bar{S}_r, \bar{S}^b_{\bar{r}})$ are linearly separable if and only if the sets $(\bar{S}''_r, \bar{S}^b_{\bar{r}})$ are linearly separable.

Algorithm 6 identifies subsets of the set $\bar{S}'_r$ that contain, in their convex hull, some unsafe state $\mathbf{u} \in$ `InseparableMinBoundaryInadmissible`, or maybe a point that dominates some unsafe state $\mathbf{u} \in$ `InseparableMinBoundaryInadmissible`. All these subsets are compiled in the set $\bar{S}''_r$. Then, it is easy to see that if the sets $(\bar{S}_r, \bar{S}^b_{\bar{r}})$ are linearly separable, the sets $(\bar{S}''_r, \bar{S}^b_{\bar{r}})$ are linearly separable as well, since, in this case, the set $\bar{S}''_r$ returned by Algorithm 6 will be the empty set. On the other hand, if the sets $(\bar{S}_r, \bar{S}^b_{\bar{r}})$ are not linearly separable, then the sets $(\bar{S}''_r, \bar{S}^b_{\bar{r}})$ are not linearly separable either, because by the construction of the set $\bar{S}''_r$ in Algorithm 6, this set contains at least one subset of $\bar{S}_r$ that has a minimal boundary inadmissible state $\mathbf{u}$ within or below its convex hull.

In view of the above two remarks, in our implementation of Algorithm 1, the branching that takes place at each node of the underlying search tree, is based on the set $\bar{S}''_r$ that corresponds to this node. This set can be considerably "thinner" than the original set $\bar{S}_r$.

The management of the list $EXPLORE$: A last issue regarding the detailing of Algorithm 1 for a tractable and streamlined execution concerns the queueing discipline to be followed in the management of the list $EXPLORE$ that is maintained by this algorithm. In our eventual implementation of Algorithm 1, this list was managed as a stack, since any other discipline resulted in an explosion of the list contents to the point that it was impos-

Algorithm 6 Calculates the state set $\bar{S}_r''$ that is used for the branching that takes place at each node of the underlying search tree

Input: $(\bar{S}_r', \bar{S}_{\bar{r}}^b)$
Output: $\bar{S}_r''$
1: $\bar{S}_r'' := \emptyset$;
2: **for all** $\mathbf{u} \in \bar{S}_{\bar{r}}^b$ **do**
3: Solve the LP

$$\min_{\mathbf{x} \geq \mathbf{0}} 0$$

s.t.

$$\begin{aligned} \sum_{k:\mathbf{s}_k \in \bar{S}_r'} x_k \mathbf{s}_k &\geq \mathbf{u} \\ \sum_{k:\mathbf{s}_k \in \bar{S}_r'} x_k &= 1 \end{aligned}$$

4: **if** the above LP is feasible **then**
5: **for all** $\mathbf{s}_k \in \bar{S}_r'$ **do**
6: **if** $x_k > 0$ **then**
7: $\bar{S}_r'' := \bar{S}_r'' \cup \{\mathbf{s}_k\}$;
8: Add the constraint $x_k = 0$ to the above LP;
9: **end if**
10: **end for**
11: GOTO step 3;
12: **end if**
13: **end for**
14: **return** $\bar{S}_r''$;

sible to maintain this list in the core memory. This fact further implies that the underlying search tree was processed according to a "depth-first" scheme.

Furthermore, according to our numerical experiments that are reported in the next subsection, a heuristic that has been very helpful in identifying high-quality policies early on in the conducted search, concerns the sequence of storing the elements of the set $\bar{S}_r''$ in the maintained stack. According to this heuristic, the elements of the set $\bar{S}_r''$ should enter the list $EXPLORE$ in a way that those elements with the largest weights in the linear combinations that are defined by the LP of Algorithm 6, should be retrieved first from the list.

3.2.3 Some numerical results

In this subsection, first we present a smaller example that provides a more concrete exposition of the various concepts and results that were introduced in this manuscript, and subsequently we report the results of a more extensive numerical experiment that seeks to (i) assess the tractability of the presented algorithm, (ii) understand (some of) the factors that might impact this tractability, and also (iii) get a more concrete view of the policy spaces $\bar{\mathcal{L}}(\Phi)$.

Applying the presented algorithm on an example RAS: In this example, we consider a conjunctive RAS Φ that supports three process types Π_1, Π_2 and Π_3. RAS Φ has six resource types – i.e., $\mathcal{R} = \{R_1, \ldots, R_6\}$ – with corresponding capacities $C_i = 4$, $i = 1, \ldots, 6$. On the other hand, the three process types Π_1, Π_2 and Π_3 that are supported by RAS Φ, possess a strictly sequential structure, and the corresponding resource allocation function $\mathcal{A}$ is described by the following vector-sequences:

$$\Pi_1 \;=\; \left\langle \begin{pmatrix} 2 \\ 0 \\ 0 \\ 0 \\ 3 \\ 0 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \\ 0 \\ 4 \\ 4 \\ 0 \end{pmatrix}, \begin{pmatrix} 3 \\ 2 \\ 0 \\ 4 \\ 2 \\ 0 \end{pmatrix}, \begin{pmatrix} 3 \\ 2 \\ 0 \\ 4 \\ 2 \\ 1 \end{pmatrix} \right\rangle$$

Table 3.2: The safe states of the example RAS in Section 3.2.3

State	State Vector	State	State Vector	State	State Vector
$\mathbf{s}_0$	(0,0,0,0,0,0,0,0,0,0,0,0,0,0)	$\mathbf{s}_{14}$	(0,0,0,0,1,1,0,0,0,0,0,0,0,0)	$\mathbf{s}_{28}$	(0,0,1,0,0,0,0,0,0,0,1,0,0,0)
$\mathbf{s}_1$	(1,0,0,0,0,0,0,0,0,0,0,0,0,0)	$\mathbf{s}_{15}$	(0,0,0,0,0,0,1,0,0,0,0,0,0,0)	$\mathbf{s}_{29}$	(0,0,0,0,1,0,0,1,0,0,0,0,0,0)
$\mathbf{s}_2$	(0,0,0,0,1,0,0,0,0,0,0,0,0,0)	$\mathbf{s}_{16}$	(0,0,0,0,0,0,0,0,0,1,1,0,0,0)	$\mathbf{s}_{30}$	(0,0,0,0,0,0,0,0,1,0,0,0,0,0)
$\mathbf{s}_3$	(0,0,0,0,0,0,0,0,1,0,0,0,0,0)	$\mathbf{s}_{17}$	(0,0,0,0,0,0,0,0,0,0,1,0,0,0)	$\mathbf{s}_{31}$	(0,0,0,0,0,0,0,0,0,1,0,0,1,0)
$\mathbf{s}_4$	(0,1,0,0,0,0,0,0,0,0,0,0,0,0)	$\mathbf{s}_{18}$	(0,0,0,1,0,0,0,0,0,0,0,0,0,0)	$\mathbf{s}_{32}$	(0,0,0,0,0,0,0,0,0,0,0,0,0,1)
$\mathbf{s}_5$	(1,0,0,0,0,0,0,0,1,0,0,0,0,0)	$\mathbf{s}_{19}$	(0,0,1,0,0,0,0,0,0,1,0,0,0,0)	$\mathbf{s}_{33}$	(0,0,0,1,0,0,0,0,0,2,0,0,0,0)
$\mathbf{s}_6$	(0,0,0,0,2,0,0,0,0,0,0,0,0,0)	$\mathbf{s}_{20}$	(0,1,0,0,0,0,0,0,0,2,0,0,0,0)	$\mathbf{s}_{34}$	(0,0,0,1,0,0,0,0,0,0,1,0,0,0)
$\mathbf{s}_7$	(0,0,0,0,0,1,0,0,0,0,0,0,0,0)	$\mathbf{s}_{21}$	(0,1,0,0,0,0,0,0,0,1,0,0,0,0)	$\mathbf{s}_{35}$	(0,0,0,0,1,0,0,0,1,0,0,0,0,0)
$\mathbf{s}_8$	(0,0,0,0,0,0,0,0,2,0,0,0,0,0)	$\mathbf{s}_{22}$	(0,0,0,0,1,0,1,0,0,0,0,0,0,0)	$\mathbf{s}_{36}$	(0,0,0,0,0,0,0,0,0,1,0,0,0,1)
$\mathbf{s}_9$	(0,0,0,0,0,0,0,0,0,1,0,0,0,0)	$\mathbf{s}_{23}$	(0,0,0,0,0,0,0,1,0,0,0,0,0,0)	$\mathbf{s}_{37}$	(0,0,0,0,0,0,0,0,0,2,0,0,0,1)
$\mathbf{s}_{10}$	(0,0,1,0,0,0,0,0,0,0,0,0,0,0)	$\mathbf{s}_{24}$	(0,0,0,0,0,0,0,0,1,0,1,0,0)	$\mathbf{s}_{38}$	(0,0,0,0,0,0,0,0,0,0,1,0,0,1)
$\mathbf{s}_{11}$	(0,1,0,0,0,0,0,0,1,0,0,0,0,0)	$\mathbf{s}_{25}$	(0,0,0,0,0,0,0,0,0,0,0,1,0)	$\mathbf{s}_{39}$	(0,0,0,0,0,0,0,0,0,0,1,1,0,0,1)
$\mathbf{s}_{12}$	(1,0,0,0,0,0,0,0,2,0,0,0,0,0)	$\mathbf{s}_{26}$	(0,0,0,1,0,0,0,0,0,1,0,0,0,0)	$\mathbf{s}_{40}$	(0,0,0,0,0,0,0,0,0,0,0,0,1,0,1)
$\mathbf{s}_{13}$	(1,0,0,0,0,0,0,0,0,1,0,0,0)	$\mathbf{s}_{27}$	(0,0,1,0,0,0,0,0,2,0,0,0,0)	$\mathbf{s}_{41}$	(0,0,0,0,0,0,0,0,0,1,0,1,0,1)

$$\Pi_2 = \left\langle \begin{pmatrix} 0\\0\\0\\2\\0\\0 \end{pmatrix}, \begin{pmatrix} 0\\1\\0\\2\\0\\4 \end{pmatrix}, \begin{pmatrix} 4\\1\\0\\2\\0\\4 \end{pmatrix}, \begin{pmatrix} 0\\1\\0\\2\\0\\4 \end{pmatrix}, \begin{pmatrix} 2\\1\\4\\2\\2\\4 \end{pmatrix} \right\rangle$$

$$\Pi_3 = \left\langle \begin{pmatrix} 0\\0\\0\\0\\0\\1 \end{pmatrix}, \begin{pmatrix} 0\\1\\0\\0\\0\\3 \end{pmatrix}, \begin{pmatrix} 0\\3\\1\\0\\0\\3 \end{pmatrix}, \begin{pmatrix} 0\\1\\1\\4\\0\\3 \end{pmatrix}, \begin{pmatrix} 3\\0\\0\\4\\0\\0 \end{pmatrix} \right\rangle$$

Hence, the state $\mathbf{s}$ of the considered RAS Φ is a vector with fourteen components: the first four components of vector $\mathbf{s}$ report the number of parts in each processing stage of the first process type; the next five components of vector $\mathbf{s}$ report the number of parts in each processing stage of the second process type; and the final five components of vector $\mathbf{s}$

report the number of parts in each processing stage of the third process type.[3] Also, state $\mathbf{s_0} = (\mathbf{0} \ldots \mathbf{0})^{\mathbf{T}}$ denotes the initial empty state.

The state space S of the considered RAS Φ consists of 98 states, from which 42 are safe. These safe states are listed in Table 3.2. Furthermore, there are 17 maximal safe states, with $\bar{S}_r = \{\mathbf{s}_6, \mathbf{s}_{12}, \mathbf{s}_{13}, \mathbf{s}_{14}, \mathbf{s}_{20}, \mathbf{s}_{21}, \mathbf{s}_{22}, \mathbf{s}_{27}, \mathbf{s}_{28}, \mathbf{s}_{29}, \mathbf{s}_{31}, \mathbf{s}_{33}, \mathbf{s}_{34}, \mathbf{s}_{35}, \mathbf{s}_{37}, \mathbf{s}_{39}, \mathbf{s}_{41}\}$, and a single boundary unsafe state that is not separable from the maximal safe states; this inseparable unsafe state has one active process instance at the first processing stage of the second process type and a second active process instance at the first processing stage of the third process type.

In order to apply the branching scheme that was presented in the previous subsection to the first node of the underlying search tree, we first try to find the maximal safe states that are extreme points of the convex hull of the set of reachable and safe states S_{rs}. It turns out that all of the 17 maximal safe states are extreme points of this convex hull, i.e., $\bar{S}'_r = \bar{S}_{rs}$. But the subsequent application of Algorithm 6 on the set $\bar{S}'_r$ returns the set $\bar{S}''_r = \{\mathbf{s}_6, \mathbf{s}_{12}\}$, and therefore, the branching process at the considered node will focus only on these two states.

Running Algorithm 5 for the child node that results from the pruning of state $\mathbf{s}_6$, we find out that the pruning of this state from the admissible space of the parent node does not require the pruning of any further nodes in order to obtain a correct policy Δ for the child node. Also, the hashset of maximal admissible states for this child node is $\texttt{MaxAdmissible} = \bar{S}_r \backslash \{\mathbf{s}_6\}$, where $\bar{S}_r$ is the set of maximal admissible states for the parent node that was reported above. On the other hand, in order to find the minimal boundary inadmissible states for this child node, we create the hashset $\texttt{PotentialMinBoundaryInadmissible} = \bar{S}^b_{\bar{r}} \cup \{s_6\}$, where $\bar{S}^b_{\bar{r}}$ denotes the set of minimal boundary inadmissible states

[3] In fact, for the purposes of deadlock avoidance that is the focus of this thesis, the state component corresponding to the last processing stage of each process type could have been dropped from the processed (state) vectors $\mathbf{s}$ without compromising the correctness of the derived policies; c.f. the corresponding discussion in [53]. But we have opted to keep these processing stages in the presented developments in order to not complicate any further the corresponding exposition of this material.

Table 3.3: The first set of the D/C-RAS configurations considered in the numerical experiment of Section 3.2.3 and the obtained results

#	Configuration			State Space				Linear DAPs					Time (sec)
	Res	Cap	Proc.	Safe	Unsafe	Max. Safe	Min. Unsafe	$\bar{S}_r''$	Count	Common	Min	Max	
1	8	4	$\{8,8,8\}$	109	1379	57	35	3	2	94	98	105	26
2	8	4	$\{7,8,8\}$	99	776	50	31	3	2	85	89	95	12
3	7	4	$\{8,8,9\}$	144	961	53	32	4	2	122	125	141	42
4	7	4	$\{8,8,9\}$	110	1692	60	37	3	2	96	100	106	101
5	7	4	$\{7,8,10\}$	128	2215	69	41	3	2	114	118	124	123
6	8	4	$\{7,8,8\}$	103	1001	54	31	4	2	89	93	99	31
7	8	4	$\{7,7,8\}$	87	490	42	29	2	2	75	77	85	2
8	7	4	$\{7,8,8\}$	103	1295	54	33	4	2	89	93	99	31
9	6	4	$\{6,8,8\}$	99	1371	49	35	3	2	84	88	95	98
10	8	4	$\{8,8,10\}$	99	1218	61	44	3	2	92	95	96	3
11	7	4	$\{8,8,8\}$	101	1461	52	28	3	2	87	91	97	23
12	6	4	$\{7,8,8\}$	139	1688	76	23	5	3	119	123	135	1062
13	8	4	$\{6,8,8\}$	97	962	51	29	4	2	77	79	85	15
14	8	4	$\{6,7,8\}$	74	455	37	27	2	2	65	66	73	2
15	7	4	$\{6,7,8\}$	105	910	59	31	5	3	83	87	101	137
16	6	4	$\{6,7,7\}$	100	906	55	31	5	3	78	82	96	31
17	7	4	$\{7,7,9\}$	104	1686	55	37	3	2	90	94	100	2
18	6	4	$\{6,7,8\}$	91	772	44	31	3	2	77	81	87	109
19	6	4	$\{7,7,8\}$	121	1481	66	21	4	3	101	105	117	1163
20	7	4	$\{7,8,8\}$	89	962	44	26	2	2	77	79	87	3
21	7	4	$\{7,7,8\}$	91	787	46	31	2	2	79	81	89	2
22	8	4	$\{5,7,8\}$	68	416	34	25	2	2	60	61	67	9
23	7	4	$\{7,7,8\}$	101	1612	53	35	3	2	87	91	97	22
24	8	4	$\{7,8,10\}$	94	1214	57	44	4	2	87	90	91	15
25	6	4	$\{6,6,7\}$	70	419	36	23	4	2	57	61	66	4

of the parent node. Finally, feeding this hashset to Algorithm 4, we find out that this hashset defines also the set of minimal boundary inadmissible states for this child node.

At this point, the algorithm proceeds to assess the linearity of the policy Δ corresponding to this child node, making use of (i) the computed sets of maximally admissible states and minimal boundary inadmissible state, and (ii) Algorithm 6. It turns out that this policy Δ is linear, indeed, and therefore, the corresponding hashset `MaxAdmissble` is entered into list $STORE$.

Next, the algorithm shifts its attention to the node that is obtained from the root node of the underlying search tree by pruning the maximal safe state $\mathbf{s}_{12}$. This node is processed similarly to the previous child node, but the resulting policy Δ' is not linear. Hence, the algorithm branches further on this node, and at the end of the algorithm execution, we find out that the two maximal linear policies for the considered RAS Φ are (a) the policy Δ that was identified in the first child node discussed above, and also (b) the policy Δ'' that admits all states of Table 3.2 except for the states in $\{\mathbf{s}_8, \mathbf{s}_{12}, \mathbf{s}_{20}, \mathbf{s}_{27}, \mathbf{s}_{33}, \mathbf{s}_{37}\}$.

A more extensive numerical experiment: In this part, we report the results of a more ex-

Table 3.4: The second set of the D/C-RAS configurations considered in the numerical experiment of Section 3.2.3 and the obtained results

#	Configuration			State Space				Linear DAPs Day1					Linear DAPs Day2			
	Res	Cap	Proc.	Safe	Uns.	Max. Safe	Min. Uns.	$\bar{S}_r''$	Cnt	Com	Min	Max	Cnt	Com	Min	Max
1	10	4	$\{8,8,8\}$	593	2756	81	32	27	3	395	399	527	3	395	399	527
2	10	4	$\{7,8,8\}$	578	2729	77	29	27	2	446	447	538	2	446	447	538
3	9	4	$\{7,7,8\}$	175	885	46	20	22	2	122	126	165	2	122	126	165
4	9	4	$\{6,8,8\}$	545	2644	68	28	25	2	440	441	506	2	440	441	506
5	7	4	$\{6,7,8\}$	491	2510	61	28	26	2	314	315	447	2	314	315	447
6	7	4	$\{6,7,7\}$	455	2160	53	25	25	2	268	269	410	2	268	269	410
7	10	4	$\{7,8,8\}$	569	2744	80	32	28	2	384	385	508	2	384	385	508
8	10	4	$\{7,7,8\}$	536	2687	71	32	27	2	356	357	495	2	356	357	495
9	9	4	$\{6,7,8\}$	521	2327	67	27	24	2	350	351	467	2	350	351	467
10	9	4	$\{7,8,8\}$	554	2550	75	31	26	2	444	445	507	2	444	445	507
11	7	4	$\{8,8,8\}$	554	2540	75	31	24	2	491	492	553	2	491	492	553
12	7	4	$\{8,8,8\}$	660	3109	84	32	24	2	539	540	587	2	539	540	587
13	7	4	$\{7,8,8\}$	634	3087	82	32	33	2	486	487	573	2	486	487	573
14	7	4	$\{7,8,8\}$	1181	6483	112	38	40	2	804	805	1073	2	804	805	1073
15	7	4	$\{6,7,8\}$	1031	5175	98	32	39	3	723	724	927	3	723	724	927
16	8	4	$\{6,8,8\}$	610	3075	81	32	31	2	501	502	554	2	501	502	554
17	8	4	$\{6,7,8\}$	577	3018	72	32	29	2	382	383	520	2	382	383	520
18	9	4	$\{6,7,7\}$	1112	4294	178	23	19	3	829	831	1043	3	829	831	1043
19	8	4	$\{6,7,7\}$	479	1947	58	24	26	3	280	282	434	3	280	282	434
20	8	4	$\{6,6,7\}$	453	1925	56	24	24	3	269	272	397	3	269	272	397
21	8	4	$\{6,7,7\}$	542	2820	67	31	25	3	411	413	497	3	411	413	497
22	7	4	$\{6,7,7\}$	479	1947	58	24	26	3	280	282	434	3	280	282	434
23	7	4	$\{5,7,8\}$	444	1749	53	23	46	3	328	331	396	3	328	331	396
24	7	4	$\{5,7,8\}$	995	5151	97	32	38	3	694	696	859	3	694	696	859
25	7	4	$\{5,7,9\}$	1682	13608	194	36	69	3	1173	1174	1499	3	1173	1174	1499

tensive computational experiment that sought to (i) assess the tractability of the proposed algorithm for computing the maximal linear DAPs on some more sizable RAS, (ii) identify potential factors that might assess this tractability, and also (iii) get a more concrete feeling of the structure of the policy spaces $\bar{\mathcal{L}}(\Phi)$. More specifically, in this experiment, we generated randomly a number of conjunctive RAS configurations, and recognizing (a) the extent of the computation to be performed by Algorithm 1, but also (b) the "off-line" nature of this computation, we gave the algorithm 48 hours to work on each configuration; if the algorithm reached this deadline of 48 hours, it was terminated and its partially obtained results were collected. The results obtained from 50 such runs are summarized in Tables 3.3 and 3.4

Table 3.3 reports the results from the algorithm executions on 25 RAS configurations Φ that were able to enumerate the corresponding sets $\bar{\mathcal{L}}(\Phi)$ within the provided time budget of 48 hours. For each of these configurations, the first two parts of Table 3.3 report (i) the size of the configuration in terms of the number of resource types involved, their capacities, the number of process types and the number of stages of each process type, and (ii) the

size and the complexity of the underlying state space as defined by the numbers of safe and unsafe states, and also the maximal safe and minimal unsafe states. The third part reports: (a) the cardinality of the set $\bar{S}''_r$ during the first iteration of the algorithm (i.e., the number of inseparable minimal boundary unsafe states that were used by the algorithm for its branching at the first level of the underlying search tree); (b) the cardinality of the computed set $\bar{\mathcal{L}}(\Phi)$ (i.e., the number of maximal linear DAPs identified by the algorithm); (c) the number of common admissible states across all these DAPs; (d) the smallest number of admissible states by a policy in $\bar{\mathcal{L}}(\Phi)$; and (e) the largest number of admissible states by a policy in $\bar{\mathcal{L}}(\Phi)$. Finally, the fourth part of Table 3.3 reports the algorithm execution time on each of these configurations.

Table 3.4 reports the results from the algorithm executions on 25 RAS configurations Φ that were not able to complete within the provided time budget of 48 hours. The first two parts of this table characterize the size and the structure of the considered RAS Φ and the underlying state spaces in the same way as in Table 3.3. On the other hand, the third and the fourth parts of Table 3.4 report (i) the cardinality of the set $\bar{S}''_r$ during the first iteration of the algorithm (similar to the corresponding column in Table 3.3), and (ii) a characterization of the contents of the list $STORE$ after (a) 24 and (b) 48 hours of the algorithm execution; this characterization is similar to the corresponding characterization that is provided for the final contents of this list in Table 3.3.

Some salient points that can be made on the basis of the contents of Tables 3.3 and 3.4 are as follows: (a) The target set $\bar{\mathcal{L}}(\Phi)$ seems to be of quite low cardinality. (b) The target policies Δ provide extensive coverage of the corresponding sets S_{rs}, and there is significant overlap among their admissible subspaces. These statements are especially obvious in the results reported in Table 3.3, but they are also corroborated by the results reported in the columns entitled 'Max' in Table 3.4. (c) A key factor that seems to determine the length of the computation of Algorithm 1 is the size of the set $\bar{S}''_r$ that is obtained by the algorithm during its first iteration, while a secondary role might be played by the cardinality of the set

$\bar{S}_{rs}$. (d) Finally, it is also interesting to notice that in the cases where the algorithm was not able to complete within the provided time interval of 48 hours, it had made almost all of its attained progress within the first 24 hours, obtaining some very good policies during this interval, while the second day apparently was spent in an effort to "validate" this initial set of results (c.f. parts 3 and 4 in Table 3.4). This realization is important because it implies that even a premature termination of the algorithm can provide some high-quality policies, and it is reminiscent of the behavior of similar search methods in the context of other combinatorial optimization problems. Finally, we should also point out that the heuristic for the management of the list $EXPLORE$ that was discussed at the end of Section 3.2.2, was instrumental for attaining this last effect.

CHAPTER 4

FLUID-RELAXATION-BASED SCHEDULING FOR CRLS

This chapter presents a scheduling methodology for the complex RAS considered in this work that is based on the solution of a "fluid" relaxation at each decision point of the original scheduling problem. The employed "fluid" relaxation for this new regime differs from the "fluid" relaxations that have been employed in past implementations of the method, since it must account for the blocking effects that take place in the considered RAS. For better clarity and specificity, the subsequent developments are focusing on the particular RAS class of the capacitated re-entrant line, and the scheduling problem addressed is the maximization of the long-term throughput of this line.

Furthermore, in the subsequent developments, the considered CRL is assumed to be controlled by a correct *linear* DAP Δ obtained, for instance, through the corresponding methodology that is presented in [59] or the developments of Chapter 3. Hence, in the context of the RAS real-time control framework of Figure 1.1, the methodology that is developed in this chapter supports essentially the function of the performance-oriented controller / scheduler in that figure.

From an organizational standpoint, the chapter consists of four major sections, with the first section introducing the proposed method itself. The second section demonstrates the application and the efficacy of the method through the example CRL of Figure 1.1. The third section provides some complexity analysis of the proposed method, highlighting the primary factors that determine this complexity, and establishing, thus, the tractability of the method. And, finally, the last section reports a series of numerical experiments that implement the presented method on a number of CRL configurations taken from some respective experiments that are reported in [39], and enable us to assess the efficacy of the method in terms of the quality of the derived solutions. At the same time, a complementary

set of experiments on larger CRL configurations also demonstrates the scalability of the presented method in terms of the involved computations.

4.1 The proposed scheduling method

The basic structure and rationale of the proposed method: As discussed in the introductory chapter, the considered scheduling method effects its decisions at each decision state reached by the underlying CRL by (i) first formulating and solving an LP that is known as the corresponding "(fluid) LP relaxation", and (ii) subsequently utilizing the information that is contained in the optimal solution of this LP in order to select an optimized action at that decision point.

The employed LP relaxation can be perceived as a simplification of the underlying scheduling problem that is obtained by treating the material processed through the considered CRL as a *continuous flow* that is constrained by the processing capacities of the line servers. Additional constraints restrict the spatial distribution of this flow across the line workstations and its time-based evolution so that it observes (i) the buffering capacities of these workstations, and (ii) the additional bounds that are imposed by the applied DAP. In this new operational setting, the considered LP seeks to maximize the line output over a predetermined time horizon T, further assuming that (a) the line workstations are initialized with the fluid levels corresponding to the current decision state $\mathbf{s}$, and (b) there exists an "infinite backlog" of fluid that can be fed into the line.

As demonstrated in the following, assuming that the employed time horizon T is sufficiently long, any optimal solution of the considered LP will seek to drive the line to a "steady-state" operational regime that maximizes its output flow rate, while minimizing the losses that will be experienced during the transient phase. This realization further suggests that the scheduler of Figure 1.1 can utilize the information about the server allocation that is encoded in the very first part of any optimal solution of the considered LP formulation, as guidance for resolving the action selection at the decision state $\mathbf{s}$ under consideration.

The detailed logic for translating the obtained solution of the LP relaxation to an action-selection scheme is the second major component of this method.

LP relaxations similar to that outlined in the previous paragraphs have been employed in the past for the scheduling of uncapacitated re-entrant lines and other more general multi-class queueing networks [73, 74, 7]. On the other hand, the particular implementation of this scheduling method in the CRL (or the more general RAS) context that is presented in this work, is differentiated from the previous instantiations of the method in the aforementioned references by (i) the integration of this scheduling methodology with the necessary logical control policies in the real-time control framework of Figure 1.1, and (ii) the ensuing need for further modification of the method in order to accommodate effectively the spatio-temporal constraints that are imposed by the workstation buffering capacities and the employed DAP.

In the rest of this section we provide all the necessary details for a complete implementation of the considered scheduling method in the CRL context. This discussion will also show that the sought integration to the employed LP relaxation of the spatio-temporal restrictions that are imposed by the workstation buffering capacities and the employed DAP, in a way that captures effectively the impact of these restrictions on the underlying operation of the line, requires some special care. One first differentiation of our implementation of the considered scheduling method compared to its past implementations in [73, 74, 7] that results from the aforementioned consideration, is the modeling of the fluidized operation of the line, and the definition of the corresponding LP formulation, in a discrete-time setting. We turn to this issue next.

Time discretization: In an effort to capture more effectively the impact of the blocking effects that are caused by the finite buffers and the imposed DAP, our LP relaxation is formulated in discrete and not in continuous time. More specifically, assuming that the mean processing times τ_j for the different processing stages $J_j,\ j = 1, \ldots, M$, are rationally valued, we set the discretizing time interval Δt equal to the greatest common divisor (GCD)

of τ_j. In this way, the mean processing time, τ_j, of any processing stage J_j, corresponds to an integral multiple of Δt, which will be denoted by $\hat{\tau}_j$. In the following discussion, we also scale time by further assuming that $\Delta t = 1.00$, and thus, $\hat{\tau}_j$ also denotes the mean processing time of processing stage J_j in this new time scale. The significance of this discretization in the context of the pursued modeling will be revealed in the detailed discussion of the employed LP formulation, which is the topic to be considered next.

The employed LP relaxation: We start the detailed presentation of the employed LP relaxation, by introducing first some supporting notation. Subsequently, we introduce the decision variables, the constraints, and the objective function, in this order.

Supporting notation:

- $\mathcal{J}_l,\ l = 1, ..., L$: The set of all processing stages executed on workstation WS_l; i.e., $\mathcal{J}_l = \{j : W(J_j) = l, j = 1, \ldots, M\}$.

- T: The total time horizon over which we are maximizing the line throughput; as explained in the opening part of this section, T is expressed in terms of the discretizing time interval Δt.

- $\mathbf{s}^{init}$: The CRL vanishing state that corresponds to the current decision state.

- $\mathbf{v}$: A $2M$-dim vector with its components $\mathbf{v}_{1+2j},\ j = 0, \ldots, M-1$, representing the volume of "fluid" waiting for the execution of processing stage J_{j+1} at the corresponding workstation $W(J_{j+1})$, and the components $\mathbf{v}_{2+2j},\ j = 0, \ldots, M-1$, representing the volume of "fluid" that has completed the execution of processing stage J_{j+1} but is still located at the corresponding workstation $W(J_{j+1})$. We shall refer to the components of vector $\mathbf{v}$ as the corresponding "fluid buffers".

- $\mathbf{v}^{init}$: The initial value for the "fluid buffer" vector $\mathbf{v}$ as defined by the state vector $\mathbf{s}^{init}$. Due to the presumed exponential nature for the distribution of the various processing times, components $\mathbf{v}_{1+2j},\ j = 0, \ldots, M-1$, will aggregate all the parts

that either wait for the initiation of the execution of the corresponding processing stage J_{j+1} or have already initiated the execution of this processing stage.

- f: A fictitious "fluid feeder" at the beginning of the line representing an "infinite backlog".

- d: A fictitious "fluid buffer" at the end of the CRL, of unlimited capacity, that collects all the "fluid" that is output by this line over the considered time horizon T.

Decision Variables:

- $x_{j,t}, j = 1, \ldots, 2M, t = 1, \ldots, T$: The "fluid" volume in "fluid buffer" $\mathbf{v}_j$ at the end of period t.

- $u_{j,t}, j = 1, \ldots, 2M+1, t = 1, \ldots, T$: The amount of the "fluid" that is added, during period t, to the "fluid buffer" $\mathbf{v}_j$ or, in the case of $j = 2M + 1$, to the "output fluid buffer" d. More specifically:

  - $u_{1,t}$ represents the amount of "fluid" that is added to the "fluid buffer" $\mathbf{v}_1$ at period t. This "fluid" is drawn from the external "fluid feeder" f, during the same period, and its addition to the "fluid buffer" $\mathbf{v}_1$ is equivalent to the action of loading new material to the CRL.

  - $u_{2+2i,t}, i = 0, ..., M - 1$, represent the amount of "fluid" that is added to the corresponding "fluid buffer" $\mathbf{v}_{2+2i}$ at period t. This "fluid" corresponds to material completing the processing of processing stage J_{i+1}, and it was drawn from "fluid buffer" $\mathbf{v}_{1+2i}$ at period $t - \hat{\tau}_{i+1} + 1$.

  - $u_{1+2i,t},\ i = 1, \ldots, M - 1$, represent the amount of "fluid" that is added to the corresponding "fluid buffer" $\mathbf{v}_{1+2i}$ at period t. This "fluid" corresponds to material transferred to this "fluid buffer" from "fluid buffer" $\mathbf{v}_{2i-1}$ during this period.

– $u_{2M+1,t}$ represents the amount of "fluid" that is transferred from the "fluid buffer" $\mathbf{v}_{2M}$ to the "output fluid buffer" d during period t.

Constraints:

1. The first set of constraints expresses the limited processing capacity at each workstation; namely, the server at each workstation cannot process more than a unit amount of work during a single time unit.

$$\sum_{j \in S_l} \sum_{q=t}^{\min\{t+\hat{\tau}_j-1,T\}} u_{2j,q} \leq 1,\ l = 1, ..., L,\ t = 1, ..., T$$

2. The second set of constraints expresses the material flow conservation; these constraints break down into the following two parts:

   (a) Material flow conservation constraints for period $t = 1$:

   $x_{1+2i,1} = \mathbf{v}_{1+2i}^{init} + u_{1+2i,1} - u_{2+2i,\hat{\tau}_{i+1}} \mathbf{1}_{\{\hat{\tau}_{i+1} \leq T\}}$,

   $i = 0 \ldots, M-1$

   $x_{2i,1} = \mathbf{v}_{2i}^{init} + u_{2i,1} - u_{1+2i,1},\ i = 1 \ldots, M$

   (b) Material flow conservation constraints for periods $t = 2, \ldots, T$:

   $x_{1+2i,t} = x_{1+2i,t-1} + u_{1+2i,t} - u_{2+2i,t+\hat{\tau}_{i+1}-1} \mathbf{1}_{\{t+\hat{\tau}_{i+1}-1 \leq T\}},\ i = 0 \ldots, M-1$

   $x_{2i,t} = x_{2i,t-1} + u_{2i,t} - u_{1+2i,t},\ i = 1 \ldots, M$

3. This set of constraints expresses the fact that a server cannot work on an empty buffer, while also acknowledging the availability of the "infinite backlog" that provides the input material for processing stage J_1; similar to the second set of constraints, we express these constraints separately for period 1 and for the remaining periods:

   (a) For period $t = 1$:

$\mathbf{v}_{1+2i}^{init} + \mathbf{v}_{2i}^{init} - u_{2+2i,\hat{\tau}_{i+1}} \mathbf{1}_{\{\hat{\tau}_{i+1} \leq T\}} \geq 0,$

$i = 1, \ldots, M-1$

(b) For periods $t = 2, \ldots, T$:

$x_{1+2i,t-1} - u_{2+2i,t+\hat{\tau}_{i+1}-1} \mathbf{1}_{\{t+\hat{\tau}_{i+1}-1 \leq T\}} \geq 0, i = 1, \ldots, M-1$

4. These constraints express the finite buffering capacity of the line workstations.

$\sum_{j \in S_l} x_{2j-1,t} + x_{2j,t} + \sum_{q=t+1}^{\min\{t+\hat{\tau}_j-1,T\}} u_{2j,q} \leq B_l, l = 1, ..., L, \;\; t = 1, ..., T$

5. These constraints account for the imposed deadlock avoidance policy Δ. The presumed linear structure of the applied DAP Δ implies that the state-admissibility condition of the policy can be expressed as a set of K inequalities having the form

$$A \cdot \hat{\mathbf{s}} \leq \mathbf{b} \tag{4.1}$$

where: (i) $\hat{\mathbf{s}}$ is the condensed state of the considered CRL, (ii) A is a $K \times M$ matrix, and (iii) $\mathbf{b}$ is a K-dim positive vector. The constraints of Equation 4.1 can be introduced in the considered LP relaxation by substituting each component $\hat{\mathbf{s}}_j, \; j = 1, \ldots, M$, of the state vector $\hat{\mathbf{s}}$ by the quantity

$$x_{2j-1,t} + x_{2j,t} + \sum_{q=t+1}^{\min\{t+\hat{\tau}_j-1,T\}} u_{2j,q}$$

6. We also want to prevent activity that will not contribute to the total output volume by the end of the time horizon T. For this, we enforce the condition that the total outflow from the network equals the total inflow to it plus the initial "fluid buffer" contents as defined by the vector $\mathbf{v}^{init}$.

$$\sum_{j=1}^{2M} \mathbf{v}_j^{init} + \sum_{t=1}^{T} u_{1,t} - \sum_{t=1}^{T} u_{2M+1,t} = 0$$

7. This constraint recognizes the fact that "fluid buffer" contents cannot be negative.

$$x_{j,t} \geq 0, \quad j = 1, ..., 2M, \;\; t = 1, ..., T$$

8. Also, the "material flows" $u_{j,t}$ cannot be negative either.

$$u_{j,t} \geq 0, \quad j = 1, ..., 2M + 1, \;\; t = 1, ..., T$$

9. Finally, the next constraint accounts for the non-preemptive nature of our scheduling policies.[1]

$$u_{2j,\hat{\tau}_j} = 1, \quad j \in \{1, \ldots, M : \mathbf{s}^{init}_{1+3(j-1)} = 1\}$$

Objective Function:

As already stated, we want to maximize the total outflow of the considered CRL over the employed time horizon T, assuming that (i) the line is operated under the relaxed modeling assumptions that are expressed by the constraints of the considered LP, and (ii) its initial "fluid buffer" contents are set to the levels that are defined by the state $\mathbf{s}^{init}$ of the original CRL model. Hence, the objective function takes the form:

$$\max \sum_{t=1}^{T} u_{2M+1,t}$$

The induced scheduling policy: After we have solved the LP relaxation, the next step is to translate the solution of the linear program to a scheduling policy for the underlying CRL. In particular, we want to use the solution of this LP as a "guide" in the selection of the next tangible state $\mathbf{s}$ among the set of tangible states that is defined by the tangible reach, $\mathcal{TR}(\mathbf{s}^{init})$, of the state $\mathbf{s}^{init}$ that constitutes the current decision point.

To effect this selection, let us denote by $\mathbf{u}^*_1$ the vector that is defined by the obtained optimal values for the variables $u_{1,1}, u_{2,\tau_1}, u_{3,1}, u_{4,\tau_2} \ldots, u_{2M,1}$, and by $\mathbf{v}$ the "fluid buffer"

[1] In the MDP formulation of Section 2.4, the non-preemptive character of the pursued policies is implied by the structure of the underlying state space S and the dynamics that are induced by this structure for that formulation.

vector that corresponds to any state $\mathbf{s} \in \mathcal{TR}(\mathbf{s}^{init})$. Then, the proposed scheduling policy will select the next tangible state, $\tilde{\mathbf{s}}$, through the following rule:

$$\tilde{\mathbf{s}} \in \arg \min_{\mathbf{s} \in \mathcal{TR}(\mathbf{s}^{init})} \sum_{j=0}^{M-1} |\mathbf{s}_{1+3j} - \mathbf{u}^*_{1,2+2j}| \tag{4.2}$$

In more natural terms, the criterion of Equation 4.2 seeks to select a tangible state $\mathbf{s} \in \mathcal{TR}(\mathbf{s}^{init})$ that has a server allocation w.r.t. the various processing stages $J_j,\ j = 1, \ldots, M$, that is most similar to the server allocation that is implied by the vector $\mathbf{u}^*_1$.

Furthermore, a secondary criterion that we have used to break any ties that are generated through the criterion of Equation 4.2, is as follows:

$$\tilde{\mathbf{s}} \in \arg \min_{\mathbf{s} \in \mathcal{TR}(\mathbf{s}^{init})} |\mathbf{v} - \mathbf{v}^{init} - \mathbf{u}^*_1|_1 \tag{4.3}$$

This new criterion perturbs the initial "buffer fluid" vector $\mathbf{v}^{init}$ by the "flow" vector $\mathbf{u}^*_1$, and eventually selects a tangible state $\mathbf{s} \in \mathcal{TR}(\mathbf{s}^{init})$ with a "fluid buffer" vector $\mathbf{v}$ that has the smallest l_1-distance from the aforementioned perturbation $\mathbf{v}^{init} + \mathbf{u}^*_1$; hence, this secondary criterion considers also state similarity in terms of buffer occupancy.

Selecting an appropriate time-horizon length T: One last parameter that needs to be further specified for the complete definition of the CRL scheduling methodology that was presented in the previous parts of this section, is the value of the parameter T to be employed in the LP relaxation, i.e., the time-horizon over which the line output will be maximized. This selection is driven by the realization that the optimal solution of the considered LP will essentially lead the system to an operational regime that provides the maximal possible output of the system as defined by the bottleneck stations of the line and the applied DAP bounds, and it will divert from this operational regime only towards the end of the operational horizon, in an effort to satisfy the termination condition of Constraint #6 above. Furthermore, the numerical experimentation that is reported in the last part of this chapter, has shown that as long as the selected T value is adequately large to let the

line reach the aforementioned operational regime, the returned vector $\mathbf{u}_1^*$ that is used in the determination of the induced scheduling policy, will be quite insensitive to the exact T value. So, with these insights and findings, we propose to set $T = (\sum_{i=1}^{L} B_i)(\sum_{j=1}^{M} \hat{\tau}_j)$, since this is an upper bound for the time that is necessary to empty the line from the entire workload that is defined by the state $\mathbf{s}^{init}$ under any globally nonidling policy.

4.2 Example

In this section, we apply the scheduling method of the previous section to the particular CRL that was considered in the example of Section 2.3 . The application of the "fluid" LP relaxation that was presented in the previous section at any of the seven vanishing states $\mathbf{s}_l,\ l \in \{12, 18, 21, 26, 33, 47, 57\}$, that constitute decision points for that CRL (c.f. Figure 2.2 and the STD of Figure 2.3), results in the following LP formulation:

$$\max_{x,u} \quad \sum_{t=1}^{T} u_{7,t}$$

s.t.

$$u_{2,t} + u_{6,t} \leq 1, \ \ t = 1, \ldots, T$$

$$u_{4,t} \leq 1, \ \ t = 1, \ldots, T$$

$$x_{1+2i,1} - u_{1+2i,1} + u_{2+2i,\hat{\tau}_{i+1}} = \mathbf{v}_{1+2i}^{init}, \ \ i = 0, \ldots, 2$$

$$x_{2i,1} - u_{2i,1} + u_{1+2i,1} = \mathbf{v}_{2i}^{init}, \ \ i = 1, \ldots, 3$$

$$x_{1+2i,t} - x_{1+2i,t-1} - u_{1+2i,t} + u_{2+2i,t+\hat{\tau}_{i+1}-1} = 0, \ \ i = 0, \ldots, 2, \ \ t = 2, \ldots, T$$

$$x_{2i,t} - x_{2i,t-1} - u_{2i,t} + u_{1+2i,t} = 0, \ \ i = 1, \ldots, 3, \ \ t = 2, \ldots, T$$

Table 4.1: Comparing the policy specified for the example CRL of Figure 2.2 by the methodology that is presented in this work to the optimal policy for this re-entrant line.

$\mathbf{s}$	Vanishing State			Tangible Reach			Opti-	Select. Crit.
	$\mathbf{s}_1\mathbf{s}_2$	$\mathbf{s}_3\mathbf{s}_4\mathbf{s}_5$	$\mathbf{s}_6\mathbf{s}_7$	$\mathbf{s}_1\mathbf{s}_2$	$\mathbf{s}_3\mathbf{s}_4\mathbf{s}_5$	$\mathbf{s}_6\mathbf{s}_7$	mal	of Eq. 4.2
12	00	100	10	10	010	10	YES	1.0626
				00	010	01	NO	1.4439
18	00	000	10	10	000	10	YES	0.7693
				00	000	01	NO	2.3220
21	00	010	10	10	010	10	YES	0.7285
				00	010	01	NO	1.2715
26	00	101	10	10	011	10	NO	1.7492
				00	010	11	YES	1.2365
33	00	001	10	10	001	10	YES	0.8282
				00	000	11	NO	1.7219
47	00	110	10	00	110	01	YES	0.6529
				10	110	10	NO	1.3471
57	00	200	10	00	110	01	YES	0.6826
				10	110	10	NO	1.5370

$$u_{2+2i,1} \leq \mathbf{v}^{init}_{1+2i} + \mathbf{v}^{init}_{2+2i}, \;\; i = 1, 2$$

$$u_{2+2i,t} - x_{1+2i,t-1} \leq 0, \;\; i = 1, 2, \;\; t = 2, \ldots, T$$

$$x_{1,t} + x_{2,t} + x_{5,t} + x_{6,t} \leq 2, \;\; t = 1, \ldots, T$$

$$x_{3,t} + x_{4,t} \leq 2, \;\; t = 1, \ldots, T$$

$$x_{1,t} + x_{2,t} + x_{3,t} + x_{4,t} \leq 3, \;\; t = 1, \ldots, T$$

$$\sum_{t=1}^{T} u_{7,t} - \sum_{t=1}^{T} u_{1,t} = \sum_{j=1}^{6} \mathbf{v}^{init}_{j}$$

$$x_{i,t} \geq 0, \;\; i = 1, \ldots, 6, \;\; t = 1, \ldots, T$$

$$u_{i,t} \geq 0, \;\; i = 1, \ldots, 7, \;\; t = 1, \ldots, T$$

$$u_{2j,\hat{\tau}_j} = 1, \;\; j \in \{1, 2, 3 : \mathbf{s}^{init}_{1+3(j-1)} = 1\}$$

The parameters $\mathbf{s}^{init}_{\cdot}$ and $\mathbf{v}^{init}_{\cdot}$ that appear in the right-hand-side of the above formulation, are determined by the considered vanishing state $\mathbf{s}_l$ according to the defining logic for these parameters that was discussed during their introduction in the earlier parts of this section.

Table 4.1 compares the policy that is defined by the solution of the above LP formulation at the seven vanishing states $\mathbf{s}_l,\ l \in \{12, 18, 21, 26, 33, 47, 57\}$, of this example CRL where $\tau_j = 1.0,\ \forall j$. Also, in line with our earlier recommendations, in the corresponding computations the parameter T was set equal to $4 \times 3 = 12$. Each primary row in Table 4.1 corresponds to one of the considered vanishing states $\mathbf{s}_l$, and the first two parts of the row provide a complete characterization of state $\mathbf{s}_l$ and its tangible reach, $\mathcal{TR}(\mathbf{s}_l)$. On the other hand, the row entry in the column entitled "Optimal" provides the choice for the next tangible state $\mathbf{s}'_l \in \mathcal{TR}(\mathbf{s}_l)$ specified by the optimal policy that is obtained through the solution of the throughput-maximizing MDP formulation for this CRL of Section 2.4. Finally, the last column of Table 4.1 provides the values for the "action"-selection criterion of Equation 4.2 that are obtained from the solution of the corresponding LP relaxations. It can be checked that, for each state $\mathbf{s}_l,\ l \in \{12, 18, 21, 26, 33, 47, 57\}$, the minimum value for this criterion corresponds to the tangible state $\mathbf{s}'_l \in \mathcal{TR}(\mathbf{s}_l)$ that is the optimal choice according to column "Optimal". Hence, for this example CRL, our scheduling methodology is able to identify an optimal policy.[2]

Figure 4.1 depicts the evolution of the variable sequences $u^*_{j,t}, t \in \{0, \ldots, 12\}$, $j = 2, 4, 6$, that constitute part of the optimal solution of the "fluid" relaxation for the example CRL of Figure 2.2 when this line is started at the vanishing state $\mathbf{s}_{12}$. In the operational context of the considered CRL, these three sequences essentially represent the server allocation that is implied by the optimal solution of the "fluid" relaxation. The three plots of Figure 4.1 exhibit clearly that the optimal solution of this relaxation drives the line to a workflow configuration where all servers maintain a constant allocation for the most part of the corresponding time horizon, except for some starting and ending phases where the solution must satisfy the specified boundary conditions. Similar behavior is exhibited for

[2]We emphasize, however, that the considered methodology does not pre-compute the applied scheduling policy in the form that is communicated in Table 4.1. At each decision state, the action that is selected by this policy is determined in real-time through (i) the formulation and solution of the corresponding LP relaxation, and (ii) the post-processing of the obtained optimal solution for this relaxation through the selection logic of Equations 4.2 and 4.3.

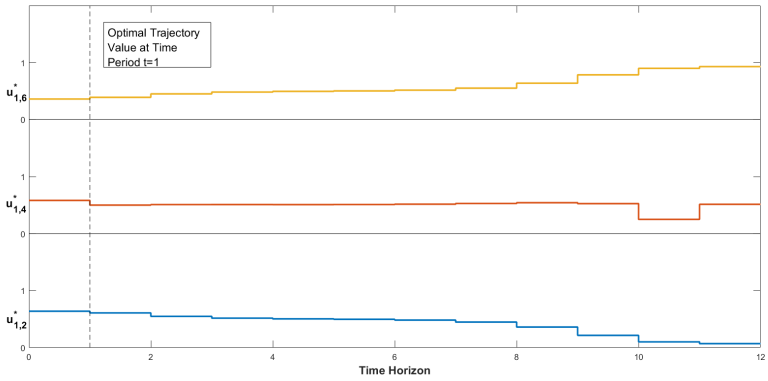

Figure 4.1: The optimal server allocation, over the entire time horizon T, that is returned by the solution of the "fluid" relaxation for the example CRL of Figure 2.2 at the vanishing state $\mathbf{s}_{12}$.

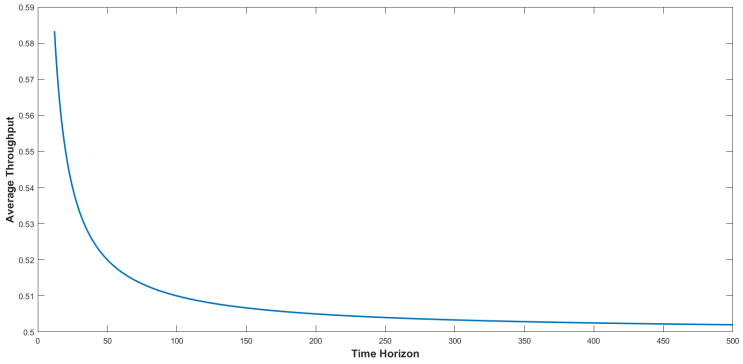

Figure 4.2: The average throughput obtained through the solution of the "fluid" relaxation for the example CRL of Figure 2.2 over different time horizons T; the starting state of the line is the vanishing state $\mathbf{s}_{12}$.

the remaining vanishing states $\mathbf{s}_l$ that constitute decision points for this line.

Also, Figure 4.2 corroborates to the above remarks, by showing that as the value of the time horizon T is increased to ever higher values, the average line throughput, under the optimal solutions of the corresponding LP formulations, converges to the value of 0.5, which is the production rate of the "bottleneck" workstation WS_1.

Finally, Figure 4.3 exhibits the server allocation for period $t = 1$ that is specified by the optimal solution of the "fluid" relaxation for the considered CRL, using a set of values for T that ranges from 12 to 500 periods. It is clearly seen in the provided plots that the

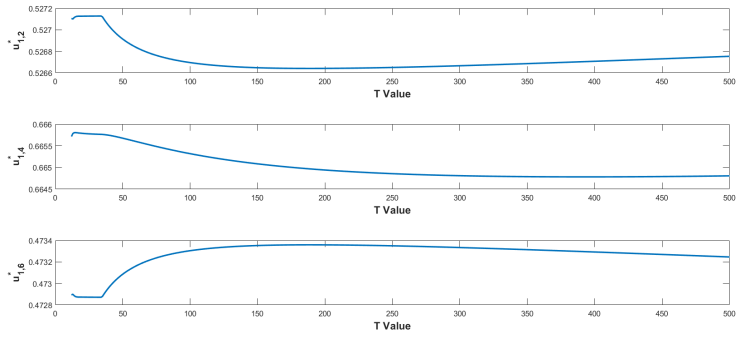

Figure 4.3: The values of the vector $\mathbf{u}_1^*$ obtained from the solution of the "fluid" relaxation for the example CRL of Figure 2.2, over different time horizons T; the starting state of the line is the vanishing state $\mathbf{s}_{12}$.

corresponding u_1^* vectors are practically insensitive to this variation of the parameter T; this fact further implies that the scheduling policy specified by the criterion of Equation 4.2 will be insensitive to this variation of T, as well.

4.3 Complexity considerations

In this section we provide some remarks that establish the tractability of the proposed scheduling method, and also reveal the primary factors that determine its computational efficiency. Also, the last part of the section outlines some additional possibilities that can be employed during the method implementation, in case that there is a need to alleviate the computational overhead that is incurred by the involved computation.

We start the overall discussion by noticing that the core of the proposed method is the solution, at each decision epoch, of the LP relaxation that was developed in Section 4.1. This LP formulation involves

- $(4M+1)T = (4M+1)(\sum_{i=1}^{L} B_i)(\sum_{j=1}^{M} \hat{\tau}_j)$ variables, and
- $(2L+2M-1+K)T + x + 1$ technological constraints.

To help the readers parse the above expressions, we also remind them that, according to the adopted notation, M stands for the number of processing stages, L stands for

the number of the line workstations, T is the length of the employed time horizon in the discretizing time unit Δt, $B_l,\ l = 1, \ldots, L$, is the buffer size at the workstation W_l, and $\hat{\tau}_j,\ j = 1, \ldots, M$, is the mean processing time of processing stage J_j under the time discretization and normalization that were introduced in Section 4.1. Also, x is the number of active servers at the current decision state $\mathbf{s}^{init}$.

Then, when we also consider the computational capabilities of current commercial LP solvers, it is clear from the above expressions, that the generated LPs will be effectively solvable by these LP solvers for a very broad spectrum of CRL configurations. This is especially true when we realize that the values $\hat{\tau}_j$ that appear in the computation of the employed time horizon T, are normalized w.r.t. the gcd of the actual mean processing times $\tau_j,\ j = 1, \ldots, M$; hence, as long as the original mean processing times do not have a very large spread, then the $\hat{\tau}_j$ values that are employed in the formulation can be pretty small.[3] The above remarks are corroborated by some experiments that are reported in the next section, and reveal that the necessary solution times for the proposed LP relaxation are in the order of a few seconds even for some pretty sizeable CRL configurations.

Yet, an additional concern arises from the fact that the considered LP relaxation must be solved "on-line" and in a repetitive fashion, i.e., at each decision point that is encountered during the real-time operation of the underlying CRL. The repetitive solution of this LP defines a computational overhead that can be significant, especially in the case that the processing times involved are pretty small, and therefore, the solution times of the formulated LPs are comparable to these processing times. If this happens to be the case, then the resulting computational overhead can be controlled through the "hashing" of the LP solutions in combination with some approximating / interpolating schemes similar to those that are outlined in [7] for the implementation of the LP-relaxation-based scheduling method that is discussed in that work; we refer the reader to [7] for some discussion on these schemes

[3] As a more vivid example of this statement, in the case of CRLs where the original mean processing times $\tau_j,\ j = 1, \ldots, M$, are equal to some constant value C, all the corresponding $\hat{\tau}_j$ will be equal to 1.0, irrespective of what is the actual value of C.

and the corresponding implementational details.

4.4 Some numerical experiments

In this section we report two series of experiments. The first set of experiments intends to further demonstrate and assess the ability of the proposed scheduling method to return scheduling policies that are (a) of comparable quality to the corresponding optimal scheduling policies, and (b) very competitive w.r.t. some other heuristic scheduling policies that are adapted from the corresponding literature. On the other hand, the second set demonstrates and assesses the computational tractability of the presented method. We organize the relevant material into two separate subsections.

4.4.1 Demonstrating and assessing the quality of the obtained schedules

In this part of the presented experiments, we used the 20 CRL configurations that are listed in Table 4.2 in order to assess the performance of the scheduling methodology that has been developed in this work against (i) the optimized performance that can be obtained (at least, in principle) through the solution of the corresponding MDP formulation of Section 2.4, and also (ii) the performance of some heuristic scheduling policies for these lines, that have been adapted from the relevant literature on the throughput maximization of uncapacitated re-entrant lines [43, 33, 18]. More specifically, for each of the CRL configurations 1 to 16 in Table 4.2, we generated 30 problem instances by varying randomly the processing rates for the corresponding processing stages over the interval [1,10]. On the other hand, for each of the CRL configurations 17 to 20 of Table 4.2, we generated only 5 problem instances, with similar ranges for the random processing rates of their processing stages, since the state spaces for these configurations are very large, and therefore, the computation of these state spaces and the performance evaluation of the corresponding scheduling policies took a very long time. Furthermore, in the employed "fluid" relaxations, we set the length of the employed time horizon $T = 20 \sum_{j=1}^{M} \hat{\tau}_j$.

Table 4.2: The CRL configurations considered in the numerical experiment of Section 4.4.1 (borrowed from [39]).

Configuration	Number Of Workstations	Number of Job Stages (JS) and Job Routes	Buffer Capacities
Conf 1	2	3JS ($W_1 \to W_2 \to W_1$)	$(B_1, B_2) = (1, 2)$
Conf 2			$(B_1, B_2) = (2, 2)$
Conf 3			$(B_1, B_2) = (3, 2)$
Conf 4			$(B_1, B_2) = (4, 4)$
Conf 5			$(B_1, B_2) = (9, 9)$
Conf 6	3	4JS($W_1 \to W_2 \to W_3 \to W_1$)	$(B_1, B_2, B_3) = (1, 2, 2)$
Conf 7			$(B_1, B_2, B_3) = (3, 2, 2)$
Conf 8			$(B_1, B_2, B_3) = (4, 3, 2)$
Conf 9			$(B_1, B_2, B_3) = (5, 5, 6)$
Conf 10	4	7JS($W_1 \to W_2 \to W_4 \to W_1 \to W_2 \to W_3 \to W_1$)	$(B_1, B_2, B_3, B_4) = (3, 2, 1, 2)$
Conf 11	3	5JS($W_1 \to W_2 \to W_3 \to W_1 \to W_2$)	$(B_1, B_2, B_3) = (3, 4, 3)$
Conf 12	3	5JS($W_1 \to W_2 \to W_3 \to W_2 \to W_3$)	$(B_1, B_2, B_3) = (3, 3, 3)$
Conf 13	3	5JS($W_1 \to W_2 \to W_1 \to W_3 \to W_2$)	$(B_1, B_2, B_3) = (3, 4, 1)$
Conf 14			$(B_1, B_2, B_3) = (2, 2, 2)$
Conf 15	3	6JS($W_1 \to W_2 \to W_3 \to W_1 \to W_2 \to W_3$)	$(B_1, B_2, B_3) = (2, 3, 2)$
Conf 16			$(B_1, B_2, B_3) = (2, 2, 2)$
Conf 17	4	7JS($W_1 \to W_2 \to W_4 \to W_1 \to W_2 \to W_3 \to W_1$)	$B_i = 3, i = 1, \ldots, 4$
Conf 18	5	7JS($W_1 \to W_2 \to W_1 \to W_3 \to W_4 \to W_5 \to W_4$)	$B_1 = B_2 = B_3 = 2$ $B_4 = B_5 = 3$
Conf 19	4	8JS($W_1 \to W_2 \to W_3 \to W_2 \to W_3 \to W_4 \to W_3 \to W_4$)	$B_i = 3, i = 1, \ldots, 5$
Conf 20	5	8JS($W_1 \to W_2 \to W_3 \to W_2 \to W_3 \to W_4 \to W_5 \to W_3$)	$B_i = 3, i = 1, \ldots, 5$

The heuristic scheduling policies that are considered in this experiment, are described as follows:[4]

[4]As already mentioned in the opening paragraph of this section, the heuristic policies that have been used as "benchmarks" for the presented experiment, constitute adaptations to the CRL operational setting of some simple policies that have been shown to be throughput-optimal for uncapacitated re-entrant lines. The performed adaptation seeks to fit the procedural logic that defines the original policies to the operational setting that is considered in this paper. But, of course, these modifications do not extend the original optimality analysis for these policies to this new setting. On the other hand, to the best of our knowledge, there are no other heuristic scheduling policies that are known to be (near-)optimal for the considered operational setting

- **"Fluid"-Relaxation(-based) Policy – FR:** This is the policy defined by the scheduling method that is developed in this work.

- **First-Buffer-First-Serve Policy– FBFS:** At any vanishing state $\mathbf{s}$ that constitutes a decision point for the underlying CRL, this policy gives priority to the state $\mathbf{s}' \in \mathcal{TR}(\mathbf{s})$ that has the line working at the earliest possible processing stage of the line. If there are many such tangible states in $\mathcal{TR}(\mathbf{s})$, the selected state is the one that incurs the largest number of part advancements from their current processing stage to the next one.

- **Last-Buffer-First-Serve Policy – LBFS:** At any vanishing state $\mathbf{s}$ that constitutes a decision point for the underlying CRL, this policy gives priority to the state $\mathbf{s}' \in \mathcal{TR}(\mathbf{s})$ that has the line working at the latest possible processing stage of the line. If there are many such tangible states in $\mathcal{TR}(\mathbf{s})$, the selected state is the one that incurs the largest number of part advancements from their current processing stage to the next one.

- **Shortest-Processing-Time-FBFS – SPT-FBFS:** At any vanishing state $\mathbf{s}$ that constitutes a decision point for the underlying CRL, this policy gives priority to the state $\mathbf{s}' \in \mathcal{TR}(\mathbf{s})$ that leads to the processing of one of the parts with the smallest expected processing time among the parts that can receive processing in the next decision epoch. In case of many such tangible states in $\mathcal{TR}(\mathbf{s})$, the final state is selected according to the FBFS logic that was defined in the first item above.

- **Shortest-Processing-Time-LBFS – SPT-LBFS:** At any vanishing state $\mathbf{s}$ that constitutes a decision point for the underlying CRL, this policy gives priority to the state $\mathbf{s}' \in \mathcal{TR}(\mathbf{s})$ that leads to the processing of one of the parts with the smallest expected processing time among the parts that can receive processing in the next decision epoch. In case of many such tangible states in $\mathcal{TR}(\mathbf{s})$, the final state is selected

and could have defined a more appropriate "benchmark" for the results that are presented in this chapter.

according to the LBFS logic that was defined in the second item above.

- **Maximum-Pressure Policy – MP:** For any state $\mathbf{s} \in S$ and any processing stage $J_j,\ j = 1, \ldots, M$, we define the "pressure" associated with processing stage J_j at state $\mathbf{s}$ as

$$\mathcal{P}(\mathbf{s}, J_j) \equiv \mu_j[(\mathbf{s}_{3(j-1)} + \mathbf{s}_{3(j-1)-1})\mathbf{1}_{\{j>1\}} +$$

$$\mathbf{s}_{1+3(j-1)} - (\mathbf{s}_{2+3(j-1)} + \mathbf{s}_{3j} + \mathbf{s}_{1+3j})\mathbf{1}_{\{j<M\}}]$$

  Also, we define the "(total) pressure" associated with state $\mathbf{s}$ by

$$\mathcal{P}(\mathbf{s}) \equiv \sum_{j=1}^{M} \mathbf{s}_{1+3(j-1)} \mathcal{P}(\mathbf{s}, J_j)$$

  i.e., $\mathcal{P}(\mathbf{s})$ is the total pressure across all processing stages that receive processing at state $\mathbf{s}$.

  Then, at any vanishing state $\mathbf{s}$ that constitutes a decision point for the underlying CRL, and for any tangible state $\mathbf{s}' \in \mathcal{TR}(\mathbf{s})$, this policy selects a state $\mathbf{s}'$ that has the highest total pressure among the states in $\mathcal{TR}(\mathbf{s})$.

The performance of each of these heuristic policies for each CRL instantiation that was generated in the considered experiment, was evaluated by solving the LP that is obtained from the corresponding Bellman equation [57] by fixing the selected actions at each decision state $\mathbf{s} \in X$ to the actions that are specified by this policy. In this way, we can characterize the level of sub-optimality for each of these policies, for any given CRL instantiation, through a "percentage (%-) error" that is defined by:

$$\%\text{-error} = \frac{\text{optimal throughput - policy throughput}}{\text{optimal throughput}} \times 100$$

Table 4.3 reports the average, minimum and maximum %-errors that were observed

during the application of the considered scheduling policies on the generated instances from the 20 CRL configurations of Table 4.2. It can be seen that the FR policy results in pretty small %-errors. This policy also outperforms the other heuristic policies in terms of, both, the average and the maximal values of these errors, a result that suggests an ability of this policy to obtain better performance than the other policies in a consistent manner. This assessment was further substantiated by performing a paired t-test [68] and a paired Wilcoxon test [75] on the %-error values that were obtained in the considered experiment. The p-values that were obtained through these two tests, assessing the dominance of the FR policy over each of the remaining heuristic policies, are reported in Table 4.4. It is clear from the values reported in this table that the difference between (a) the %-errors attained by the FR policy and (b) the %-errors that are attained by the other scheduling policies, is statistically significant. This finding further implies that the "fluid" relaxation developed in this work, and the accompanying logic of Equations 4.2 and 4.3, manage to capture effectively the basic workflow dynamics of the considered CRLs that shape their performance.

Finally, we also report that the largest LP formulations resulting from the "fluid" relaxation of the CRL configurations of Table 4.2 were solved in less than 3 seconds. Hence, unless the scale of the processing times involved is very small (i.e., of the order of a few seconds), the presented methodology will be very comfortably implemented in the context of the CRLs that are considered in this experiment. But in the next section, we also consider more explicitly the scaling of the LP solution times as the size of the underlying CRL increases.

4.4.2 Demonstrating and assessing the tractability of the presented method

In this section we report and discuss the results from an additional set of numerical experiments that intended to investigate empirically the increase of the solution time of the proposed LP relaxation as the underlying CRLs are scaled up to some pretty sizable con-

Table 4.3: An empirical characterization of the performance that is attained by the various heuristic policies considered in the experiment of Section 4.4.1.

Config.	% error	FR	FBFS	LBFS	SPT-FBFS	SPT-LBFS	MP
Conf 1	Avg.	0	2.323356	0	1.049778	0.581273	0
	Min.	0	0.230038	0	0	0	0
	Max.	0	7.453339	0	3.863308	3.486496	0
Conf 2	Avg.	0	2.563791	0	1.24137	0.765513	0
	Min.	0	0.02166	0	0	0	0
	Max.	0	7.803391	0	4.640675	3.682091	0
Conf 3	Avg.	0.558299	4.281953	0.986135	2.446898	2.3089	1.580789
	Min.	0	0.00912	0	0.00048	0	0.00112
	Max.	1.604902	12.173708	2.566975	5.517447	4.551116	3.45924
Conf 4	Avg.	0.424129	2.53043	0.116397	0.893047	0.868761	1.627846
	Min.	0	0	0	0	0	0
	Max.	1.603027	11.14905	0.503845	3.707549	3.707549	6.765885
Conf 5	Avg.	0.056252	1.072059	0.000941	0.250837	0.189375	0.845842
	Min.	0	0	0	0	0	0
	Max.	0.627573	8.386695	0.011211	1.843072	1.248081	5.951457
Conf 6	Avg.	0.787126	3.087116	0.424329	1.501644	1.074064	2.058273
	Min.	0	0.045081	0.000827	0.00506	0.000827	0.01904
	Max.	2.468721	10.00878	1.711332	6.370564	4.164482	7.04788
Conf 7	Avg.	0.262069	2.921857	2.242041	2.497875	2.538068	1.514924
	Min.	0	0.0143	0.009941	0.0143	0.0147	0.045611
	Max.	1.077797	9.432003	7.589988	8.149082	7.843766	5.124153
Conf 8	Avg.	0.348257	2.601589	1.797056	2.174541	2.152668	2.067119
	Min.	0.00012	0.00948	0.00022	0.00948	0.01296	0.013749
	Max.	1.365613	12.276351	9.255655	7.782629	7.782629	8.948562
Conf 9	Avg.	0.074912	1.331306	0.341868	0.668099	0.640542	0.586824
	Min.	0	0	0	0	0	0
	Max.	0.353171	9.699296	2.322959	4.391537	4.391537	5.283458
Conf 10	Avg.	3.318123	5.666601	4.888519	5.035612	4.983614	4.061828
	Min.	0.160619	0.781178	0.372858	0.954592	0.94859	0.445644
	Max.	6.94834	14.570786	12.520645	15.677988	15.684631	9.899285
Conf 11	Avg.	0.837663	2.777581	3.677889	1.303101	1.69863	1.633319
	Min.	0.032066	0.017783	0.257924	0.017783	0.092037	0.161028
	Max.	1.800238	12.551314	10.558961	4.213472	7.117571	4.232176
Conf 12	Avg.	1.121402	1.750244	2.452877	0.513728	0.583221	2.630212
	Min.	0.008642	0.032489	0.001541	0.001401	0.001541	0.019846
	Max.	2.579237	6.878605	8.388345	0.984445	1.650506	8.486197
Conf 13	Avg.	1.153944	2.264382	4.354457	2.158219	2.663196	2.176783
	Min.	0.034693	0.056632	0.222106	0.158161	0.2213	0.030272
	Max.	2.823261	8.684196	10.984664	8.385657	9.550511	5.152178
Conf 14	Avg.	1.330487	4.610859	1.528115	2.281134	2.089135	3.060204
	Min.	0.009692	0.047781	0.058834	0.058834	0.059004	0.099643
	Max.	2.752433	8.670543	4.278931	4.618316	5.159253	5.3891
Conf 15	Avg.	1.087259	2.441627	3.636966	0.92349	1.102124	2.274138
	Min.	0.484144	0.448128	1.277105	0.143185	0.143185	0.662133
	Max.	2.075733	9.279909	10.152834	3.754415	3.754415	5.461083
Conf 16	Avg.	1.485586	2.141188	3.186278	0.876134	1.005724	2.443547
	Min.	0.077542	0.025837	0.43741	0.025837	0.025837	0.195422
	Max.	3.165834	7.171983	8.032857	2.690495	2.823853	4.837758
Conf 17	Avg.	1.23399	3.190605	6.283274	3.711618	4.010187	3.144512
	Min.	0	0.490636	0.030681	0.490636	0.030681	0.067602
	Max.	3.581841	6.302036	15.98632	7.48842	7.48842	7.174882
Conf 18	Avg.	3.431575	8.666107	10.35568	8.993217	8.655773	11.74234
	Min.	1.294327	6.396949	6.432713	6.396949	6.432713	7.038907
	Max.	4.468809	11.394275	16.312115	12.035385	10.939199	16.799726
Conf 19	Avg.	3.372908	8.528441	10.312104	8.869568	8.533083	11.517266
	Min.	1.07888	5.816134	6.021595	5.816134	5.638849	7.232241
	Max.	4.546285	10.8824	16.257885	11.561644	10.519218	16.014564
Conf 20	Avg.	3.35003	8.658793	10.693207	9.087125	8.751012	11.758839
	Min.	0.77972	5.625051	8.035321	6.695957	6.205683	7.809635
	Max.	5.313402	12.155388	16.182583	12.155388	11.16347	17.40793

Table 4.4: A statistical comparison of the performance of the proposed scheduling methodology to the performance of the other heuristic policies considered in the experiment of Section 4.4.1.

Method	FBFS	LBFS	SPT-FBFS	SPT-LBFS	MP
t-test	4.405229e-48	1.071112e-16	4.142378e-12	5.152820e-11	9.354007e-22
w-test	5.339791e-52	6.755720e-15	4.400029e-14	1.003275e-12	2.285927e-35

figurations. These results are reported in Table 4.5.

More specifically, the considered LP relaxation was formulated and solved for 20 con-

Table 4.5: An empirical characterization of the computational tractability of the proposed scheduling method.

Basic configuration	# of passes	LP sol. time (sec) (min, mean, max)
5 workstations	2 (10 stages)	(0.23, 0.27, 0.34)
$B_l = 5,\ \forall l$	3 (15 stages)	(0.34, 0.37, 0.42)
$\tau_j = 1,\ \forall j$	4 (20 stages)	(0.46, 0.50, 0.59)
	5 (25 stages)	(0.71, 0.82, 0.91)
	6 (30 stages)	(0.93, 1.06,1,34)
5 workstations	2 (10 stages)	(7.71, 8.16, 8.74)
$B_l = 5,\ \forall l$	3 (15 stages)	(15.53, 16.51, 17.76)
$\tau_1 = 1$	4 (20 stages)	(26.52, 27.51, 28.42)
$\tau_j = 10,\ \forall j \neq 1$	5 (25 stages)	(36.33, 40.14, 44.22)
	6 (30 stages)	(59.98, 59.29, 60.72)
20 workstations	2 (40 stages)	(0.47,0.52,0.70)
$B_l = 5,\ \forall l$	3 (60 stages)	(1.30, 1.46, 1.71)
$\tau_j = 1,\ \forall j$	4 (80 stages)	(1.67, 1.80, 2.05)
	5 (100 stages)	(2.42, 2.55, 2.69)
	6 (120 stages)	(2.81, 3.12, 3.42)
20 workstations	2 (40 stages)	(31.46, 34.25, 38.20)
$B_l = 5,\ \forall l$	3 (60 stages)	(56.94, 68.72, 80.39)
$\tau_1 = 1$	4 (80 stages)	(95.39, 99.58, 106.38)
$\tau_j = 10,\ \forall j \neq 1$	5 (100 stages)	(136.19, 142.29, 151.85)
	6 (120 stages)	(189.25, 197.34, 203.49)

figurations, with each configuration being defined by the first two columns of Table 4.5. In particular, the first column of this table reports the number of workstations of the corresponding configurations, the buffer size of each workstation, and the mean processing time of the involved processing stages. On the other hand, the second column of Table 4.5 reports the number of times that the line is traversed by each part, and therefore it also determines the number of stages of the corresponding process plan. Finally, the third column of Table 4.5 reports the solution times for the corresponding LP relaxations; for each of the 20 CRL configurations, the LP relaxation was formulated and solved at 10 randomly selected decision states, and this column of Table 4.5 reports the minimum, maximum and average values of the corresponding solution times. We also note that the formulated LPs were solved through the CPLEX Studio IDE 12.8 package, running on a Windows 10 computational platform with an Intel Core i5, 2.2 GHz, 2-core processor, and 8GB DDR3

memory.

As seen in Table 4.5, the solution times for the proposed LP relaxation can be at the order of a few seconds even for some pretty large configurations. This fact is especially true as long as the "spread" of the mean processing times involved is quite small; the corresponding cases are those in blocks #1 and #3 of Table 4.5.

On the other hand, the results of Table 4.5 also suggest that the considered solution times can be severely impacted by a larger "spread" among the underlying processing times. This can be seen by juxtaposing the results of block #2 to those of block #1, and similarly, the results of block #4 to those of block #3. In particular, the configurations of block #4 in Table 4.5 correspond to a "worst-case" scenario where the underlying CRL is pretty sizeable in terms of numbers of workstations and processing stages, and at the same time, all mean processing times involved have very large values except for one. Then, as indicated by the provided formulae in Section 4.3, the resulting LPs will be pretty large in terms of the numbers of variables and constraints involved, and therefore, their solution times might scale up to the order of a few minutes. Whether these solution times will be tolerable or not, will depend on the magnitude of the actual processing times involved. If these times are also pretty small, then it might be necessary to control further the computational overhead that is incurred by the presented method, by employing some of the mechanisms that were suggested in the closing part of Section 4.3. Furthermore, in this case, it is also pertinent to try to control more carefully the length of the time horizon T that is employed by the relaxing LP. Some mechanisms that can facilitate a better control of this parameter are discussed in the next chapter.

CHAPTER 5
AN IMPLEMENTATION OF THE FR-BASED SCHEDULING METHOD THROUGH TIMED-CONTINUOUS-PETRI-NET-BASED MODELING AND ANALYSIS

This chapter intends to show that the fluid-relaxation-based scheduling method that was developed in Chapter 4 can be enhanced by leveraging the modeling and analytical power of timed-continuous Petri nets (tc-PNs) [44]. More specifically, the fluid relaxation model and the corresponding relaxing LP that are used in the implementation of the FR-based scheduling method in Chapter 4, have been developed through some *ad hoc* representations and arguments. This chapter intends to show that timed-continuous Petri nets provide a natural and more structured medium for the representation of these fluid relaxations, and that the existing theory for the tc-PN model also enables (i) a more systematic derivation of the relaxing LP, and (ii) an analytical study of the solution space of this LP and the structure of its optimal solutions. Besides their theoretical interest, these results further enable (iii) a more informed parameterization of the relaxing LP, and (iv) a systematic extension of the methodology to RAS with very complex structure and dynamics. Finally, the tc-PN-based representations that are employed in this chapter, have also enabled us to identify certain structures and conditions in the operation of the original GSPN that can be a source of confusion for the FR-based scheduling methodology that is pursued in this work.

This chapter is organized as follows: Section 5.1 abstracts the operation of the considered CRL in the modeling framework of Generalized Stochastic PNs (GSPNs) [1], and overviews the representation of linear DAPs in this framework through the theory of "monitor" places [23, 48]. Section 5.2 overviews the definition and some basic properties of untimed and timed continuous PN models, and subsequently it uses this material in order to define a fluidized version of the original CRL of Section 5.1 as a timed continuous PN,

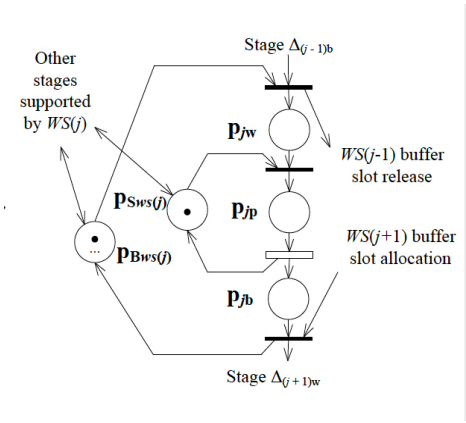

Figure 5.1: The GSPN subnet modeling a single processing stage of the considered CRL in the GSPN modeling framework.

and a corresponding throughput-maximization problem in this tc-PN model. On the other hand, Section 5.3 uses the developments of Section 5.2 in order to detail the FR-based scheduling method for the considered CRL, and for the underlying GSPN model that is presented in this chapter. Section 5.4 briefly discusses the extension of this approach to the broader class of the D/C-RAS of Definition 1. Finally, Section 5.5 presents the identified cases where the FR-based scheduling method might result in suboptimal choices.

5.1 Modeling the considered CRL as a GSPN

In this section, we introduce an alternative formal model for the structure and the operation of the CRL model that was introduced in Section 2.3.1, that is based on the GSPN modeling framework. In the subsequent discussion, we shall assume that the reader is familiar with the basic PN modeling framework. Furthermore, Appendix A overviews some basic concepts and properties of this framework that are necessary for the developments that are presented in this work, and introduces the relevant notation that will be employed in the rest of this chapter.

Modeling the CRL workflow as a PN: The workflow materialized by the considered CRL model can be formally represented by a PN that has the particular structure of a

"process-resource" net [59]. More specifically, this PN will consist of (i) a "process" subnet that will model the sequential logic of the single process plan that is supported by the considered CRL, and (ii) a set of "resource" places that will model the availability of the various buffers and servers that are utilized for the support of the various processing stages of the aforementioned process plan, and the allocation of these resource units to the executing process instances.

Figure 5.1 depicts the subnet of the aforementioned process-resource net that models the dynamics of the j-th processing stage, J_j, of the considered CRL. More specifically, places p_{jw}, p_{jp} and p_{jb} and their interconnecting transitions model the part of the aforementioned process subnet that models processing stage J_j. Tokens in place p_{jw} represent process instances waiting for the execution of processing stage J_j, at workstation $WS(J_j)$; tokens in place p_{jp} represent process instances executing processing stage J_j, at workstation $WS(J_j)$; and tokens in place p_{jb} represent process instances that have completed the execution of processing stage J_j but are still located at workstation $WS(J_j)$. In the following, we shall denote these three substages respectively by J_{jw}, J_{jp} and J_{jb}. Furthermore, assuming that the considered CRL starts idle and empty of any jobs, the initial marking $\mathbf{m}_0(p_{jx})$ of all places p_{jx}, $j = 1, \ldots, M,\ x \in \{w, p, b\}$, must be set equal to zero.

The places $p_{B_{WS}(j)}$ and $p_{S_{WS}(j)}$ depicted in Figure 5.1 are "resource" places. More specifically, tokens in place $p_{B_{WS}(j)}$ model the free buffer slots at the workstation $WS(j)$; as indicated in the figure, a process instance entering this workstation for the execution of processing stage J_j must be allocated one of the free buffer slots, and this buffer slot will be released when this process instance leaves the workstation, heading to the workstation that will support the execution of the next processing stage, J_{j+1} (or exits the CRL, if processing stage J_j is the last processing stage). The initial marking of place $p_{B_{WS}(j)}$ is $\mathbf{m}_0(p_{B_{WS}(j)}) = B_{WS(j)}$, i.e., the buffering capacity of workstation $WS(j)$. Place $p_{S_{WS}(j)}$ models the server availability at workstation $WS(j)$. The tokens in this place are required only for the execution of the corresponding substage J_{jp}. Furthermore, since we assume

single-server workstations, we shall also have $\mathbf{m}_0(p_{S_{WS(j)}}) = 1$.

For the needs of the subsequent developments, it is also important to notice that each CRL-modeling PN $\mathcal{N}$ possesses the following minimal p-semiflows that are defined respectively by the finite buffering and processing capacity at each workstation $WS_i,\ i = 1, \ldots, L$:

$$\mathbf{m}(p_{B_{WS_i}}) \ + \sum_{j:WS(j)=WS_i} \Big(\mathbf{m}(p_{jw}) + \mathbf{m}(p_{jp}) + \mathbf{m}(p_{jb}) \Big) = B_i \tag{5.1}$$

$$\mathbf{m}(p_{S_{WS_i}}) \ + \sum_{j:WS(j)=WS_i} \mathbf{m}(p_{jp}) = 1 \tag{5.2}$$

Finally, the next proposition establishes some important properties of the CRL-modeling PN $\mathcal{N}$ that will be useful in the following developments.

Proposition 5 *The CRL-modeling PN $\mathcal{N}$ possesses the following properties:*

1. *$\mathcal{N}$ is conservative, consistent and structurally bounded.*
2. *$|R(\mathcal{N}, \mathbf{m}_0)| < \infty$, $\forall\ \mathbf{m}_0$.*
3. *$\mathcal{N}$ is a mono-t-semiflow net.*
4. *Under the initial marking $\mathbf{m}_0$ that was specified in the previous paragraphs, the considered net $\mathcal{N}$ is quasi-live.*

Proof: In order to prove the conservative nature of net $\mathcal{N}$, consider the summation of Equations 5.1 and 5.2 for all workstations $WS_i,\ i = 1, \ldots, L$. This summation results in the p-semiflow

$$\sum_{i=1}^{L} \left[\mathbf{m}(p_{B_{WS_i}}) + \mathbf{m}(p_{S_{WS_i}}) + \sum_{j:WS(j)=WS_i} \Big(\mathbf{m}(p_{jw}) + 2 \cdot \mathbf{m}(p_{jp}) + \mathbf{m}(p_{jb}) \Big) \right] = \sum_{i=1}^{L} (B_i + 1) \tag{5.3}$$

which includes the marking of every place $p \in P$ of net $\mathcal{N}$; hence, $\mathcal{N}$ is conservative.

The conservative nature of $\mathcal{N}$ subsequently implies its structural boundedness. Also, Item #2 in Proposition 5 is an immediate consequence of the structural boundedness of net $\mathcal{N}$.

On the other hand, in order to establish the consistency and the mono-t-semiflow property of net $\mathcal{N}$, let us denote the four transitions appearing in the corresponding path of Figure 5.1 by $t_{j1},\ t_{j2},\ t_{j3},\ t_{j+1,1}$. Then, it can be easily checked that the transition sequence $\sigma = t_{11}t_{12}t_{13}t_{21}\ldots t_{M1}t_{M2}t_{M3}t_{M+1,1}$ corresponds to the complete processing of a single process instance through the underlying CRL. Clearly, the Parikh vector of σ is $|\sigma| = \mathbf{1}$, where $\mathbf{1}$ is a $|T|$-dim vector with all its components equal to one. Furthermore, letting Θ denote the flow matrix of net $\mathcal{N}$, we shall have $\Theta\cdot|\sigma| = 0$. Hence, $\mathcal{N}$ is consistent.

The reader can also check that the aforementioned sequence σ is the only minimal t-semiflow of $\mathcal{N}$. Since $\mathcal{N}$ is also consistent and conservative, it is a mono-t-semiflow net.

Finally, the transition sequence σ that is mentioned in the previous paragraphs is also feasible under the initial marking $\mathbf{m}_0$ of $\mathcal{N}$ that was specified in the previous paragraphs. Since $|\sigma| = \mathbf{1}$, the net $\mathcal{N}$ is quasi-live under this initial marking. $\square$

Example: Figure 5.2 concretizes the CRL-modeling PN structure that was defined in the previous paragraphs, by providing the PN that models the CRL of Figure 2.2. The path $\langle t_0, p_0, \ldots, p_6, t_7\rangle$ models the process subnet of this PN. Places p_8 and p_{10} model respectively the buffer availability for workstations WS_1 and WS_2, and places p_7 and p_9 model the server availability for these two workstations.

However, for a complete understanding of the PN of Figure 5.2, the reader must also notice that places p_{1w} and p_{3b} have been dropped from the corresponding process subnet. This simplification is justified by the working assumption of zero loading and unloading times for the considered CRL model, and it will be presumed in all the technical developments and the supporting examples that will be presented in the following.

Finally, the p-semiflows of Equations 5.1 and 5.2 for the PN model of Figure 5.2 are as

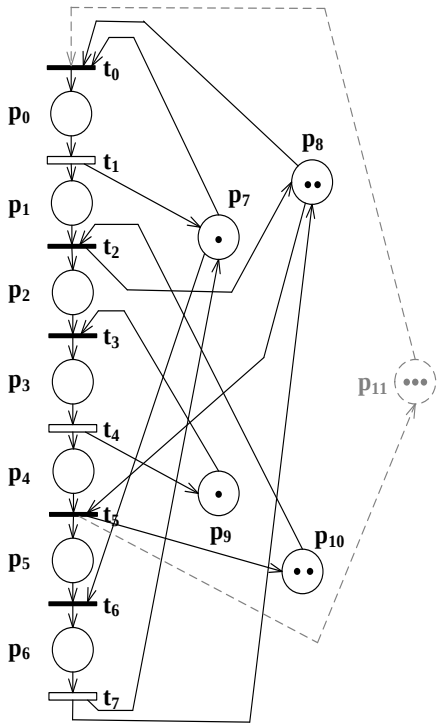

Figure 5.2: The GSPN model for the CRL depicted in Figure 2.2.

follows:

$$\mathbf{m}(p_8) + \mathbf{m}(p_0) + \mathbf{m}(p_1) + \mathbf{m}(p_5) + \mathbf{m}(p_6) = 2 \tag{5.4}$$

$$\mathbf{m}(p_{10}) + \mathbf{m}(p_2) + \mathbf{m}(p_3) + \mathbf{m}(p_4) = 2 \tag{5.5}$$

$$\mathbf{m}(p_7) + \mathbf{m}(p_0) + \mathbf{m}(p_6) = 1 \tag{5.6}$$

$$\mathbf{m}(p_8) + \mathbf{m}(p_3) = 1 \tag{5.7}$$

Liveness-enforcing supervision of the considered PN model: In the PN-based representation of the considered CRL model, the requirement for deadlock-free resource allocation translates naturally to a requirement for reversibility of the corresponding process-

resource net. Furthermore, the work presented in [62, 63] establishes the following important result:

Proposition 6 *In a class of process-resource nets that subsumes the CRL-modeling PN $\mathcal{N}$ that was defined in the previous paragraphs, reversibility is equivalent to liveness.*

Example: In the PN of Figure 5.2, the deadlock states of the underlying CRL that were discussed in Section 2.3.2, are represented by any feasible marking $\mathbf{m}$ satisfying

$$\mathbf{m}(p_0) + \mathbf{m}(p_1) + \mathbf{m}(p_2) + \mathbf{m}(p_3) + \mathbf{m}(p_4) = 4 \tag{5.8}$$

These markings can be effectively blocked from the net dynamics, in a minimally restrictive manner, by imposing the inequality:

$$\mathbf{m}(p_0) + \mathbf{m}(p_1) + \mathbf{m}(p_2) + \mathbf{m}(p_3) + \mathbf{m}(p_4) \leq 3 \tag{5.9}$$

The above inequality can be enforced upon the plant PN of Figure 5.2 by super-imposing upon this net the subnet that is depicted in dashed lines in that figure. This subnet introduces the "monitor" place p_{11} with initial marking $\mathbf{m}_0(p_{11}) = 3$, and it can be systematically synthesized through the corresponding theory of [48]. From a functional standpoint, the "monitor" place p_{11} limits the token flow in the process subnet defined by the path $\langle t_0, p_0, \ldots, p_6, t_7 \rangle$ in a way that is not very different from the limitation that is exerted upon this token flow by the various resource places of the line; therefore, we can perceive this "monitor" place as an additional "(fictitious) resource" place for the underlying PN model. And this "fictitious resource" place introduces an additional minimal p-semiflow in the dynamics of the controlled PN, with a structure similar to that of the p-semiflows that are defined by Equations 5.1 and 5.2. □

Clearly, the enforcement of the inequality of Equation 5.9 on the dynamics of the PN of Figure 5.2 through the superimposition of the corresponding "monitor" place p_{11}, as dis-

cussed in the previous example, can be extended to any set of similar linear inequalities that define a correct DAP. Furthermore, the resulting PN that models the dynamics of the logically controlled CRL, remains in the class of process-resource nets. Hence, the employed PN modeling framework of process-resource nets constitutes a very effective platform for the integrative representation of, both, the plant RAS and the necessary DAP.

Modeling the timed dynamics of the considered CRL model: The CRL-modeling PN and the corresponding problem of liveness-enforcing supervision that were introduced in the previous parts of this section, concern the characterization and the study of all the possible event sequences w.r.t. process initiation, advancement and completion that can take place in the underlying CRL. In the relevant DES terminology [9], these event sequences constitute the "untimed" dynamics of the considered CRL and of the corresponding PN model.

On the other hand, the "timed" dynamics of this CRL can be systematically characterized and investigated by extending the original PN model to a GSPN [1]. This is attained by associating an exponential distribution with each transition of the original PN model that represents an event with non-zero execution times. In the considered CRL model, the only such events are the events that correspond to the processing of a part at some particular processing stage. Furthermore, in the processing-stage-modeling PN subnet that is depicted in Figure 5.1, the corresponding event is represented by the output transition of place p_{jp}; hence, this transition is characterized as a "timed" transition of the corresponding GSPN model, and its firing times are drawn from an exponential distribution with instantaneous rate μ_j, to be denoted $\mathcal{E}xp(\mu_j)$. Also, in the typical GSPN semantics, the timed nature of the aforementioned transition is indicated by depicting this transition by a white rectangle (and this is also what has been adopted in Figures 5.1 and 5.2).

The remaining transitions are characterized as "untimed", and they are represented by a black bar. Since untimed transitions require zero time for their execution, they can fire

spontaneously, as soon as they are enabled, and therefore, in standard GSPN theory, they are presumed to have precedence over any timed transitions that are also enabled at a given marking.

Finally, for further reference, in the following discussion we shall denote the set of the timed transitions of the considered GSPN models by T_t, and the complementary set of untimed transitions by T_u.

The aforestated assumptions regarding the firing dynamics of the timed and the untimed transitions of any given GSPN model $\mathcal{N}$ induce a partitioning of the underlying reachability space $R(\mathcal{N}, \mathbf{m}_0)$ into "vanishing" and "tangible" markings; the respective subspaces will be denoted by $R_V(\mathcal{N}, \mathbf{m}_0)$ and $R_T(\mathcal{N}, \mathbf{m}_0)$. For a formal definition of these two subspaces, consider a marking $\mathbf{m} \in R(\mathcal{N}, \mathbf{m}_0)$, and let $\mathcal{E}(\mathbf{m})$ denote the set of the enabled transitions at $\mathbf{m}$. Then, $\mathbf{m} \in R_V(\mathcal{N}, \mathbf{m}_0)$ *iff* $\mathcal{E}(\mathbf{m}) \cap T_u \neq \emptyset$; otherwise, $\mathbf{m} \in R_T(\mathcal{N}, \mathbf{m}_0)$.

The exponential and independent nature of the firing times of the timed transitions of net $\mathcal{N}$ further implies that each marking $\mathbf{m} \in R_T(\mathcal{N}, \mathbf{m}_0)$ will have an exponentially distributed sojourn time that is defined by the "exponential race" [9] of the set of its enabled transitions $\mathcal{E}(\mathbf{m})$. Furthermore, this exponential race also determines the firing probability $p(t; \mathbf{m})$ of each enabled transition $t \in \mathcal{E}(\mathbf{m})$ at marking $\mathbf{m}$. On the other hand, for a vanishing marking $\mathbf{m} \in R_V(\mathcal{N}, \mathbf{m}_0)$ that enables more than one untimed transition, there is a need for an extraneous specification of a probability distribution that will arbitrate the firing of these untimed transitions. In the standard GSPN terminology, this externally specified probability distribution is known as a "random switch", and in the following it will be denoted by $rs(\mathbf{m})$.

The timed dynamics of a GSPN model $\mathcal{N}$ with a well defined set of random switches for all the vanishing markings $\mathbf{m} \in R_V(\mathcal{N}, \mathbf{m}_0)$, constitutes a semi-Markov process [67], to be denoted by $\mathcal{SM}(\mathcal{N})$. In the case of the GSPN $\mathcal{N}$ that models the operation of a CRL under the control of a correct linear DAP, the necessary random switches can be specified in a way that the resulting semi-Markov process $\mathcal{SM}(\mathcal{N})$ is ergodic. Then, there will

exist a "stationary" probability distribution $\psi(\mathcal{N})$, defined on the set of tangible markings $R_T(\mathcal{N}, \mathbf{m}_0)$,[1] that will characterize the "long term" (or "steady state") behavior of the underlying CRL, and will enable the definition and the computation of similar performance measures for this system.

As explained in Chapter 2, in this work we are especially interested in such a set of random switches that maximizes the long-term throughput of the underlying CRL. Furthermore, the sought random switches can be obtained, in principle, through an MDP formulation that is obtained from a straightforward translation of the MDP formulation of Section 2.4 to the GSPN semantics that were defined in the previous paragraphs.

5.2 Fluidization of the GSPN model $\mathcal{N}$

One way to overcome the computational challenges that arise from the explosive size of the state space of the MDP corresponding to the GSPN model $\mathcal{N}$, while retaining a significant part of the structure of the corresponding scheduling problem, is by considering an approximation of this problem that is defined through a "fluidization" of the original GSPN model $\mathcal{N}$. The resulting system is an alternative PN model that is known as a "timed continuous PN (tc-PN)". This new PN model possesses the same net structure with the original GSPN $\mathcal{N}$, but it adheres to modified semantics and dynamics for the net marking, that will enable the characterization of the underlying state space, and the various levels of (steady-state) throughput that can be attained in this state space, through efficient algebraic methods.

Hence, in the first part of the section we overview the definition of tc-PNs, and present some properties of these models that are useful for the derivation of the results that are pursued in this paper. The second part of the section focuses on the particular tc-PN that is derived from the fluidization of the CRL-modeling GSPN $\mathcal{N}$, to be denoted by $\mathcal{N}^{(tc)}$, and investigates the problem of maximizing the steady-state throughput for this particular model. The results of this section will be utilized eventually in Section 5.3 in order to

[1] since vanishing markings have zero sojourn times

develop a heuristic scheduling method for the throughput maximization of the original GSPN $\mathcal{N}$.

5.2.1 Untimed and Timed Continuous Petri nets

Untimed Continuous Petri nets: An untimed continuous PN (uc-PN) system $(\mathcal{N}, \mathbf{m}_0)$ is a PN system with the net structure $\mathcal{N}$ being defined as in the case of the classical PN system (c.f. Definition 15 in Appendiix A), but with the marking $\mathbf{m}$ of the net taking nonnegative *real* values (i.e., $\mathbf{m} : P \rightarrow \mathbb{R}_0^+$).

In the semantics of the uc-PN system, a transition $t \in T$ is enabled at some marking $\mathbf{m}$ *iff* $\forall p \in {}^\bullet t,\ \mathbf{m}(p) > 0$. For any enabled transition t at $\mathbf{m}$, we further define the "enabling degree" $enab(t, \mathbf{m})$ as follows:[2]

$$enab(t, \mathbf{m}) = \min_{p \in {}^\bullet t} \left\{ \frac{\mathbf{m}(p)}{\Theta^-(p, t)} \right\} \tag{5.10}$$

At a given marking $\mathbf{m}$, an enabled transition t can fire at any amount $0 \leq a \leq enab(t, \mathbf{m})$, leading to a new marking

$$\mathbf{m}' = \mathbf{m} + a \cdot \Theta(\cdot, t) \tag{5.11}$$

where $\Theta(\cdot, t)$ denotes the column of the flow matrix Θ corresponding to transition t. This new definition of transition fireability in uc-PNs further implies that a fireable transition sequence σ at some marking $\mathbf{m}$ is completely specified not only by the transition sequence itself, but also by the level of firing of each of these transitions at the corresponding marking. With this understanding, we define the reachability space $R^{uc}(\mathcal{N}, \mathbf{m}_0)$ of a uc-PN

[2] We remind the reader that the matrix Θ^- that appears in Equation 5.10, is the "pre-flow" matrix of net $\mathcal{N}$, that encodes the restriction of the flow relation W of Definition 15 on the $(P \times T)$ part of its domain. Please, refer to Appendix A for further details.

system $(\mathcal{N}, \mathbf{m}_0)$ as follows:

$$R^{uc}(\mathcal{N}, \mathbf{m}_0) = \left\{\mathbf{m} | \exists \sigma = a_1 t_1 \dots a_k t_k \text{ s.t. } \mathbf{m}_0[\sigma\rangle \mathbf{m}\right\} \tag{5.12}$$

But while $R^{uc}(\mathcal{N}, \mathbf{m}_0)$ collects all the markings $\mathbf{m}$ that are reachable from the initial marking $\mathbf{m}_0$ through the firing of a finite transition sequence σ, in uc-PNs we can also have markings that are obtained in the limit, through the firing of an infinite transition sequence. This possibility is captured by the notion of "lim-reachability", which is formally defined as follows:

Definition 12 *[58] In a uc-PN system $(\mathcal{N}, \mathbf{m}_0)$, a marking $\mathbf{m}$ is* lim-reachable *iff there exists a sequence of reachable markings $\{\mathbf{m}_i\}_{i \geq 1}$ such that*

$$\mathbf{m}_0[\sigma_1\rangle \mathbf{m}_1[\sigma_2\rangle \mathbf{m}_2 \dots \mathbf{m}_{i-1}[\sigma_i\rangle \mathbf{m}_i \dots$$

and $\lim_{i \to \infty} \mathbf{m}_i = \mathbf{m}$. Furthermore, the marking set that results from the augmentation of the reachability set $R^{uc}(\mathcal{N}, \mathbf{m}_0)$ with the lim-reachable markings of $(\mathcal{N}, \mathbf{m}_0)$ will be denoted by lim–$R^{uc}(\mathcal{N}, \mathbf{m}_0)$.

An important property of *lim–*$R^{uc}(\mathcal{N}, \mathbf{m}_0)$ is that it is a closed convex set [58]. Furthermore, the notions of boundedness, liveness, quasi-liveness, reversibility and deadlock extend naturally over the reachability space $R^{uc}(\mathcal{N}, \mathbf{m}_0)$. In addition, [58] introduces the following notions of "lim-liveness" and "lim-deadlock":

Definition 13 *In an uc-PN system $(\mathcal{N}, \mathbf{m}_0)$, a transition t is* lim-live *iff*

$$\forall \mathbf{m} \in \textit{lim–}R^{uc}(\mathcal{N}, \mathbf{m}_0), \exists \mathbf{m}' \in R^{uc}(\mathcal{N}, \mathbf{m}) \textit{ s.t. } enab(t, \mathbf{m}') > 0$$

Furthermore, a uc-PN system $(\mathcal{N}, \mathbf{m}_0)$ lim-deadlocks *iff*

$$\exists \mathbf{m} \in \textit{lim–}R^{uc}(\mathcal{N}, \mathbf{m}_0) \textit{ s.t. } \forall t \in T,\ enab(t, \mathbf{m}) = 0$$

Finally, there are structural versions for boundedness, liveness and lim–liveness, that are defined in the same spirit as in the original PN system of Appendix A.

Since the structure of a uc-PN system $(\mathcal{N}, \mathbf{m}_0)$ is defined by the same net structure $\mathcal{N}$ that is employed in Appendix A, this new class of PNs inherits all the structural concepts and properties that have been defined for the original PNs in that section. Of particular interest in the subsequent discussion are the notions of "p- and t-semiflows" of these nets, that are respectively defined as the nonnegative left and right annulers of the flow matrix Θ. In view of Equations 5.11 and 5.12, these two types of semiflows have similar connotations for the dynamics of the uc-PNs with the connotations of their counterparts for the dynamics of the original PNs of Appendix A. They also induce the same notions of "consistency" and "conservativeness"' as in the original PNs, with similar implications for the dynamics of the underlying uc-PN system. Furthermore, in the case of uc-PNs, the notion of a "mono-T-semiflow net" is defined in exactly the same manner as in the original PNs (c.f. Definition 17 in Appendix A). Finally, the following proposition collects some properties of uc-PNs that will be useful in the subsequent developments; formal proofs for these results can be traced in [58, 69]

Proposition 7 *If a uc-PN system* $(\mathcal{N}, \mathbf{m}_0)$ *is consistent and quasi-live, then*

$$\mathbf{m} \in \textit{lim–}R^{uc}(\mathcal{N}, \mathbf{m}_0) \iff \exists \sigma \geq \mathbf{0} \textit{ s.t. } \mathbf{m} = \mathbf{m}_0 + \Theta \cdot \sigma$$

Furthermore, if system $(\mathcal{N}, \mathbf{m}_0)$ *is also conservative, then the above two statements are also equivalent to the following one:*

$$\forall\ p\textit{-semiflow } \mathbf{y},\ \mathbf{y} \cdot \mathbf{m} = \mathbf{y} \cdot \mathbf{m}_0$$

Timed Continuous Petri nets: Next we introduce the class of timed continuous PNs (tc-PNs) and overview some results for this class that are necessary for this work. tc-PNs are obtained from the class of uc-PNs, by associating a firing rate $\mu(t)$ with every transition $t \in T$. Then, adopting the "infinite-server" semantics for this class of nets,[3] we define the "flow vector" $\mathbf{f}(\mathbf{m})$, for the transitions $t \in T$ at some marking $\mathbf{m}$, as follows:

$$\forall t \in T, \;\; \mathbf{f}(t;\mathbf{m}) = \mu(t) \cdot enab(t,\mathbf{m}) = \mu(t) \cdot \min_{p \in {}^\bullet t} \left\{ \frac{\mathbf{m}(p)}{\Theta^-(p,t)} \right\} \tag{5.13}$$

From a more conceptual standpoint, the flow vector $\mathbf{f}(\mathbf{m})$ defines the rate of change of marking $\mathbf{m}$ when the net transitions are fired at their maximum possible speed; for each transition $t \in T$, this maximum speed is defined by the firing rate $\mu(t)$ and the enabling degree of this transition at marking $\mathbf{m}$. Hence, if we let τ denote the absolute time, and we consider the net marking $\mathbf{m}$ as a function of τ, we shall have that (c.f. Equation 5.11)

$$\mathbf{m}(\tau) = \mathbf{m}_0 + \Theta \cdot \sigma(\tau) \tag{5.14}$$

and differentiating both sides of this equation w.r.t. τ, we get

$$\dot{\mathbf{m}}(\tau) = \Theta \cdot \dot{\sigma}(\tau) = \Theta \cdot \mathbf{f}(\mathbf{m}(\tau)) \tag{5.15}$$

In the subsequent developments, we shall further allow that $\mu(t) = \infty$ for some transitions $t \in T$, which will enable us to replicate the notion of "untimed" transitions of the GSPN modeling framework in the tc-PN context. It is clear from Equation 5.13, that for these transitions, $\mathbf{f}(t;\mathbf{m}) = \infty$, for all markings $\mathbf{m}$, and therefore, these transitions can fire instantaneously at marking $\mathbf{m}$ at any level that does not exceed $enab(t,\mathbf{m})$ (i.e., their enabling degree at that marking).

[3] The adoption of the "infinite-server" semantics for tc-PNs in this work is justified by the explicit modeling of the resources that regulate the firing of the various transitions of the CRL-modeling PN $\mathcal{N}$ through the corresponding "resource" places; the reader is referred to Sections 2.3.1 and 3.1 of [44] for a more thorough support of this statement.

In addition, we shall consider a "controlled" version of tc-PNs, where

$$\forall t \in T,\ \ \mathbf{f}(t;\mathbf{m}) \leq \mu(t) \cdot enab(t,\mathbf{m}) = \mu(t) \cdot \min_{p \in {}^{\bullet}t} \left\{ \frac{\mathbf{m}(p)}{\Theta^{-}(p,t)} \right\} \tag{5.16}$$

i.e., transitions $t \in T$ can be "slowed down" w.r.t. their maximal firing speeds. Controlled tc-PNs have similar reachability dynamics to the underlying uc-PNs, and they inherit the structural and behavioral properties of the latter.

A last concept that we must introduce in the modeling regime of tc-PNs before we turn to the particular tc-PN structure that corresponds to the CRL-modeling GSPN $\mathcal{N}$, is that of the "steady-state operation" of a controlled tc-PN. For the needs of this work, we define this concept as follows:

Definition 14 *A controlled tc-PN* can be operated at a steady-state regime *at some marking* $\mathbf{m}$ *iff*

$$\exists\, \mathbf{f} \in \mathbb{R}^{|T|} \textit{ s.t. } \Big(\forall t \in T,\ \mathbf{0} < \mathbf{f}(t;\mathbf{m}) \leq \mu(t) \cdot enab(t,\mathbf{m}) \Big) \ \wedge \ \Big(\Theta \cdot \mathbf{f} = \mathbf{0} \Big)$$

Marking $\mathbf{m}$ *itself is characterized as* a potential steady-state marking.

In other words, a given marking $\mathbf{m}$ of tc-PN $\mathcal{N}$ is a potential steady-state marking, if there is a strictly positive t-semiflow of $\mathcal{N}$ that constitutes a feasible flow vector $\mathbf{f}$ for marking $\mathbf{m}$ under the controlled dynamics of $\mathcal{N}$. Then, marking $\mathbf{m}$ will remain unaltered under the firing of the net transitions according to the considered flow vector $\mathbf{f}$. Furthermore, the strict positivity of $\mathbf{f}$ implies that this operation will keep active the entire network.

Next we turn our attention to the dynamics of the particular ct-PN that is induced by the fluidization of the CRL-modeling GSPN $\mathcal{N}$ that was introduced in Section 5.1, focusing primarily on the potential steady-state markings of this net.

5.2.2 A fluidized version for the CRL-modeling GSPN $\mathcal{N}$

In this subsection, we (i) abstract the CRL-modeling GSPN $\mathcal{N}$ to a tc-PN $\mathcal{N}^{(tc)}$; (ii) establish some basic properties for this new PN system; and (iii) characterize the space of the potential steady-state markings of $\mathcal{N}^{(tc)}$ and the maximal flow that can be attained at such a marking. The results and the insights to be developed in this subsection will be at the core of the scheduling method for the original CRL-modeling GSPN $\mathcal{N}$ that will be developed in Section 5.3.

The considered net $\mathcal{N}^{(tc)}$: It should be obvious from the definitions and the discussion that were provided in the previous subsection, that the tc-PN $\mathcal{N}^{(tc)}$ which is the fluidized version of the CRL-modeling GSPN $\mathcal{N}$ of Section 5.1, can be obtained straightforwardly from the latter by (i) letting the marking $\mathbf{m}$ of this net attain nonnegative real values, and (ii) re-interpreting the firing rates, μ_t, of its timed transitions $t \in T_t$, according to the semantics that are defined by Equations 5.14–5.16.

Furthermore, net $\mathcal{N}^{(tc)}$ inherits all the properties that were established in Proposition 5 of Section 5.1 for the original PN $\mathcal{N}$ that was introduced in that section. Also, the minimal p-semiflows of net $\mathcal{N}^{(tc)}$ consist of (i) those p-semiflows that are defined by Equations 5.1–5.2 in Section 5.1, and (ii) the p-semiflows that that are induced by the "monitor" places that implement the applied linear DAP Δ.[4] And since $\mathcal{N}^{(tc)}$ is consistent, quasi-live and conservative (according to Proposition 5), Proposition 7 further implies that these minimal p-semiflows also provide a complete characterization of the lim-reachability space, *lim–*$R^{uc}(\mathcal{N}, \mathbf{m}_0)$, for the untimed fluidized version of the original PN $\mathcal{N}$ of Section 5.1. In order to further enable this characterization, in the following, all minimal p-semiflows of the considered net $\mathcal{N}^{(tc)}$ will be collected in the rows of a matrix to be denoted by $B_y(\mathcal{N})$.

Finally, since the considered tc-PN $\mathcal{N}^{(tc)}$ is a controlled tc-PN w.r.t. the firing rate of its timed transitions, the lim-reachability space *lim–*$R^{uc}(\mathcal{N}, \mathbf{m}_0)$ constitutes also the lim-

[4] c.f. the corresponding discussion in the example of Section 5.1.

reachability space for net $\mathcal{N}^{(tc)}$ itself.

Characterizing the potential steady-state markings of $\mathcal{N}^{(tc)}$: Next, we employ all the above results for the tc-PN $\mathcal{N}^{(tc)}$ in order to characterize the space of its potential steady-state markings, and the connectivity of this space to the overall lim-reachability space, *lim–*$R^{uc}(\mathcal{N}, \mathbf{m}_0)$.

As discussed in the proof of Proposition 5, the unique minimal t-semiflow $\mathbf{x}$ for the CRL-modeling PN $\mathcal{N}$, and, therefore, for the considered tc-PN $\mathcal{N}^{(tc)}$, is the $|T|$-dim vector $\mathbf{1}$ (i.e., the vector with all its components equal to 1). This further implies that any flow vector $\mathbf{f}$ that can define a steady-state regime at any marking $\mathbf{m} \in$ lim–$R^{uc}(\mathcal{N}, \mathbf{m}_0)$, will have the following structure for some scalar $f > 0$:

$$\mathbf{f} = f \cdot \mathbf{1} \tag{5.17}$$

Furthermore, according to the definition of the CRL-modeling GSPN $\mathcal{N}$ in Section 5.1, the set of timed transitions, T_t, for this net consists of the transitions t_j that are the output transitions for the places p_{jp}, $j = 1, \ldots, M$, that model the job processing at the corresponding processing stage J_j (c.f., Figure 5.1). And for any timed transition $t_j \in T_t$, ${}^\bullet t_j = \{p_{jp}\}$ with $\Theta^-(p_{jp}, t_j) = 1$.

Then, in view of Definition 14 and the second part of Proposition 7, we have the following characterization of the space of the potential steady-state markings of the tc-PN $\mathcal{N}^{(tc)}$:

Proposition 8 *The set of potential steady-state markings for the tc-PN $\mathcal{N}^{(tc)}$ is given by:*

$$SS(\mathcal{N}^{(tc)}) \equiv \left\{ \mathbf{m} \in \mathbb{R}_0^+ \;\middle|\; \Big(B_y(\mathcal{N}) \cdot \mathbf{m} = B_y(\mathcal{N}) \cdot \mathbf{m}_0 \Big) \wedge \Big(\forall j = 1, \ldots, M,\ \mathbf{m}(p_{jp}) > 0 \Big) \right\}$$

Proof: Proposition 8 is an immediate consequence of the remarks in the paragraphs that precede this proposition, upon noticing the following two additional facts:

First, Proposition 7 implies that the set

$$\left\{ \mathbf{m} \in \mathbb{R}_0^+ \;\middle|\; B_y(\mathcal{N}) \cdot \mathbf{m} = B_y(\mathcal{N}) \cdot \mathbf{m}_0 \right\} \tag{5.18}$$

is a complete characterization of the lim-reachability space of the considered net $\mathcal{N}^{(tc)}$. This fact explains the first part of the condition that is posed upon the sought markings $\mathbf{m}$ by Proposition 8.

On the other hand, the sought markings must further satisfy the condition of Definition 14. But in view of (i) Equation 5.17, (ii) the untimed transitions that are present in $\mathcal{N}^{(tc)}$, and (iii) the inflow relation of the timed transitions $t_j,\ j = 1, \ldots, M$, of the net to their corresponding input places p_{jp}, the condition of Definition 14 reduces to the following condition:[5]

$$\exists\, f \in \mathbb{R}^+ \text{ s.t. } \forall t_j \in T_t,\ f \leq \mu(t_j) \cdot \mathbf{m}(p_{jp}) \tag{5.19}$$

The proof concludes by noticing that the second part in the condition that is posed upon the sought markings $\mathbf{m}$ by Proposition 8, is essentially a rewriting of the condition of Equation 5.19. □

In the following, we are particularly interested in those elements of the marking set $SS(\mathcal{N}^{(tc)})$ that will enable a maximal steady-state flow f^* for the underlying tc-PN $\mathcal{N}^{(tc)}$. In view of all the previous discussion, this set of markings, and the corresponding maximal flow f^*, can be obtained as optimal solutions to the following linear program (LP):

$$\max f \tag{5.20}$$

[5]We remind the reader that $\mathbb{R}^+$ denotes the set of strictly positive reals.

s.t.

$$B_y(\mathcal{N}) \cdot \mathbf{m} = B_y(\mathcal{N}) \cdot \mathbf{m}_0 \tag{5.21}$$

$$\forall t_j \in T_t,\ f \leq \mu(t_j) \cdot \mathbf{m}(p_{jp}) \tag{5.22}$$

$$f \geq 0 \tag{5.23}$$

The reader should notice that the required non-negativity for the vector variable $\mathbf{m}$ that appears in the above LP, is enforced by the combination of Equations 5.22 and 5.23 and the strict positivity of the rates $\mu(t_j),\ \forall t_j \in T_t$.

On the other hand, the existence of an optimal solution $(f^*, \mathbf{m}^*)$ for this LP with $f^* > 0$ is guaranteed by (i) the resource availability that is presumed by the CRL model that is considered in this work, and (ii) the logical correctness of the applied DAP Δ; these two elements define the effective content of Equation 5.21.

Finally, for further reference, we shall denote the set of markings $\mathbf{m}^*$ that constitute part of an optimal solution $(f^*, \mathbf{m}^*)$ for the LP of Equations 5.20–5.23, by $OSS(\mathcal{N}^{(tc)})$.

Example: For the tc-PN $\mathcal{N}^{(tc)}$ that is induced from the GSPN $\mathcal{N}$ of Figure 5.2, the LP of Equations 5.20–5.23 can be written as follows:

$$\max f \tag{5.24}$$

s.t.

$$\mathbf{m}(p_0) + \mathbf{m}(p_1) + \mathbf{m}(p_5) + \mathbf{m}(p_6) \leq 2 \tag{5.25}$$

$$\mathbf{m}(p_2) + \mathbf{m}(p_3) + \mathbf{m}(p_4) \leq 2 \tag{5.26}$$

$$\mathbf{m}(p_0) + \mathbf{m}(p_6) \leq 1 \tag{5.27}$$

$$\mathbf{m}(p_3) \leq 1 \tag{5.28}$$

$$\mathbf{m}(p_0) + \mathbf{m}(p_1) + \mathbf{m}(p_2) + \mathbf{m}(p_3) + \mathbf{m}(p_4) \leq 3 \tag{5.29}$$

$$f \leq \mu_1 \cdot \mathbf{m}(p_0) \tag{5.30}$$

$$f \leq \mu_2 \cdot \mathbf{m}(p_3) \tag{5.31}$$

$$f \leq \mu_3 \cdot \mathbf{m}(p_6) \tag{5.32}$$

$$f \geq 0 \tag{5.33}$$

Next, we provide a more intuitive interpretation of the content of this LP regarding the maximal levels of flow f that are attainable in the corresponding tc-PN $\mathcal{N}^{(tc)}$. For a start, the reader should notice that the structure of the above LP implies that there will exist an optimal solution $(f^*, \mathbf{m}^*)$ with

$$\mathbf{m}^*(p_1) = \mathbf{m}^*(p_2) = \mathbf{m}^*(p_4) = \mathbf{m}^*(p_5) = 0 \tag{5.34}$$

When we add the constraint of Equation 5.34 to the LP of Equations 5.24–5.33, the constraints of Equations 5.25 and 5.26 of that LP become redundant, and we obtain the

following simplified LP:

$$\max f \tag{5.35}$$

s.t.

$$\mathbf{m}(p_0) + \mathbf{m}(p_6) \leq 1 \tag{5.36}$$

$$\mathbf{m}(p_3) \leq 1 \tag{5.37}$$

$$\mathbf{m}(p_0) + \mathbf{m}(p_3) \leq 3 \tag{5.38}$$

$$f \leq \mu_1 \cdot \mathbf{m}(p_0) \tag{5.39}$$

$$f \leq \mu_2 \cdot \mathbf{m}(p_3) \tag{5.40}$$

$$f \leq \mu_3 \cdot \mathbf{m}(p_6) \tag{5.41}$$

$$f \geq 0 \tag{5.42}$$

Furthermore, it is easy to see that, in this new LP, the constraint of Equation 5.38 is also redundant since it is implied by the constraints of Equations 5.36 and 5.37.

Finally, the LP that is defined by the remaining six constraints can be transformed into the following form:

$$\max f \tag{5.43}$$

s.t

$$f \cdot (\frac{1}{\mu_1} + \frac{1}{\mu_3}) \leq 1 \tag{5.44}$$

$$f \cdot (\frac{1}{\mu_2}) \leq 1 \tag{5.45}$$

$$f \geq 0 \tag{5.46}$$

This last LP implies that, in the considered tc-PN $\mathcal{N}^{(tc)}$, the maximal flow f^* is defined by the processing capacities at the two single-server workstations WS_1 and WS_2 of the underlying CRL.

However, it is also interesting to notice that if the buffering capacities B_1 and B_2 for the two workstations WS_1 and WS_2 in the considered example were equal to one unit instead of two, then, the right-hand-side of Equation 5.38, that represents the corresponding maximally permissive LES, would be equal to 1+1-1 = 1. Hence, the constraint that is defined by this equation in the LP of Equations 5.35–5.42 would not be redundant, and, therefore, the LP of Equations 5.43–5.46 should be augmented with the additional constraint

$$f \cdot (\frac{1}{\mu_1} + \frac{1}{\mu_2}) \leq 1 \tag{5.47}$$

From a more conceptual standpoint, the constraint of Equation 5.47 can be perceived as a potential "virtual bottleneck" [19] for the tc-PN $\mathcal{N}^{(tc)}$ that is defined by the applied DAP.

□

We close this section with a result that establishes the finite accessibility of the set $OSS(\mathcal{N}^{(tc)})$ from any marking $\mathbf{m} \in$ *lim–*$R^{uc}(\mathcal{N}, \mathbf{m}_0)$, of the considered net $\mathcal{N}^{(tc)}$.

Proposition 9 *Let f^* denote the maximal steady-state flow for the considered net $\mathcal{N}^{(tc)}$, and define the marking $\tilde{\mathbf{m}}$ of this net as follows:*

$\forall p \in P,\ \tilde{\mathbf{m}}(p) =$

$$\begin{cases} \frac{f^*}{\mu(t_j)}, & \textit{if } p = p_{jp} \textit{ for some } j \in \{1, \ldots, M\} \\ 0, & \textit{if } p = p_{jw} \textit{ or } p_{jb} \textit{ for some } j \in \{1, \ldots, M\} \\ \mathbf{m}_0(p) - \sum_{p' \in P \setminus \{p\}} \mathbf{y}_p(p') \cdot \tilde{\mathbf{m}}(p'), & \textit{for every place } p \textit{ modeling a resource or} \\ & \textit{a monitor place with corresponding} \\ & p\textit{-semiflow } \mathbf{y}_p \end{cases} \tag{5.48}$$

Then, marking $\tilde{\mathbf{m}}$ *satisfies the following two properties:*

1. $\tilde{\mathbf{m}} \in OSS(\mathcal{N}^{(tc)})$.

2. $\tilde{\mathbf{m}} \in R^{uc}(\mathcal{N}, \mathbf{m}),\ \forall \mathbf{m} \in$ *lim–*$R^{uc}(\mathcal{N}, \mathbf{m}_0)$.

Proof: In order to prove the first part of Proposition 9, it suffices to show that marking $\tilde{\mathbf{m}}$ satisfies the constraints of the LP of Equations 5.20–5.23, for $f = f^*$. The constraint of Equation 5.22 is satisfied immediately by the definition of marking $\tilde{\mathbf{m}}$. Furthermore, since f^* is the maximal steady-state flow of net $\mathcal{N}^{(tc)}$, there exists a marking $\mathbf{m}^*$ such that $(f^*, \mathbf{m}^*)$ is an optimal solution to the LP of Equations 5.20–5.23. Also, Equations 5.22 and 5.48 imply that $\tilde{\mathbf{m}}(p) \leq \mathbf{m}^*(p), \forall p \in \{p_{jw}, p_{jp}, p_{jb} : j = 1, \ldots, M\}$. Therefore,

$$\mathbf{m}_0(p) - \sum_{p' \in P \setminus \{p\}} \mathbf{y}_p(p') \cdot \tilde{\mathbf{m}}(p') \ \geq\ \mathbf{m}_0(p) - \sum_{p' \in P \setminus \{p\}} \mathbf{y}_p(p') \cdot \mathbf{m}^*(p') \ \geq\ 0$$

and marking $\tilde{\mathbf{m}}$ satisfies the constraints of Equation 5.21, as well.

Next, we shall establish the second part of Proposition 9 by providing a finite transition sequence $\sigma = \sigma_1\sigma_2\sigma_3\sigma_4$ that will lead from any given marking $\mathbf{m} \in$ lim–$R^{uc}(\mathcal{N}, \mathbf{m}_0)$ to the target marking $\tilde{\mathbf{m}}$.

Transition subsequence σ_1 will first establish a "corridor" of free capacity $\delta < \min\{\tilde{\mathbf{m}}(p_{jp}) : j = 1, \ldots, M\}$ w.r.t. each resource or monitor place q across the entire line. This can be attained in an iterative manner, starting from the place p_{Mp} and unloading the

necessary amount of fluid from this place in order to attain the aforestated free-capacity requirement w.r.t. any resource or monitor place q that includes place p_{Mp} in the corresponding p-semiflow. Subsequently, we employ the free capacity established through this fluid removal from place p_{Mp}, in order to satisfy the same free-capacity requirement w.r.t. the resource and the monitor places that are engaged by the places $p_{M-1,w},\ p_{M-1,p},\ p_{M-1,b}$, that model stage J_{M-1}; we omit the relevant details to the reader. Sequence σ_1 is completed by proceeding in a similar manner through the remaining stages $J_{M-2}, \ldots, J_1$, in this order. Let $\mathbf{m}'$ denote the marking that will result from this draining.

Transition subsequence σ_2 will employ the "corridor of free capacity" established by sequence σ_1 in order to drain the line from the following quantities:

- For each place $p \in \{p_{jp} | j = 1, \ldots, M\}$ with $\mathbf{m}'(p) > \tilde{\mathbf{m}}(p) - \delta$, transition subsequence σ_2 will remove an amount of fluid equal to $\mathbf{m}'(p) - \tilde{\mathbf{m}}(p) + \delta$.

- For each place $p \in \{p_{jw}, p_{jb} | j = 1, \ldots, M\}$ with $\mathbf{m}'(p) > \tilde{\mathbf{m}}(p) = 0$, transition subsequence σ_2 will empty completely this place by removing the corresponding amount of fluid $\mathbf{m}'(p)$.

For all these places, the corresponding drainage will occur in chunks no larger than δ. Then, since the original marking $\mathbf{m}$ satisfies all the minimal p-semiflows of net $\mathcal{N}$, all the markings that will be generated by the fluid advancement through the line during this drainage will satisfy these p-semiflows, as well (i.e., they will respect the resource availability of the line and the imposed LES). Let us denote the marking that will result from the execution of transition subsequence σ_2 by $\mathbf{m}''$.

Transition subsequence σ_3 will add to places $p \in \{p_{jp} : j = 1, \ldots, M\}$ with $\mathbf{m}''(p) < \tilde{\mathbf{m}}(p) - \delta$, the quantities $\tilde{\mathbf{m}}(p) - \delta - \mathbf{m}''(p)$. For each such place p, the corresponding quantity will be loaded from the beginning of the line, in chunks no larger than δ. Let us denote the marking that will result from the execution of transition subsequence σ_3 by $\mathbf{m}'''$.

Finally, transition subsequence σ_4 will bring to each place $p_{jp},\ j = 1, \ldots, M$, a fluid

amount equal to δ, starting with place p_{Mp}, and proceeding with places $p_{M-1,p}, \ldots, p_{1p}$, in this order. The plausibility of this operation w.r.t. the p-semiflows of net $\mathcal{N}$ is guaranteed by the specification of (i) marking $\tilde{\mathbf{m}}$ in Equation 5.48, and (ii) the intermediate target markings $\mathbf{m}''$ and $\mathbf{m}'''$. $\square$

Next, we use the result of Proposition 9 in order to develop a "fluid relaxation" for the CRL scheduling problem that is considered in this work, in the CRL-modeling framework of the tc-PN $\mathcal{N}^{(tc)}$ that was defined in this section.

5.3 The proposed scheduling method

As discussed in Chapter 4, the "fluid relaxation (FR)"-based method to the CRL throughput-maximization problem that is considered in this work, tries to determine a near-optimal scheduling policy in real-time, according to an on-line computational scheme that, at each decision state, operates as follows:

1. First, it defines and solves an LP formulation that is known as a "fluid relaxation" of the original problem; this LP formulation constitutes a continuous, more tractable abstraction of the underlying resource allocation process, while the initial condition of this process is set to the current decision state.

2. Once the aforementioned LP has been solved, the proposed scheduler employs the information that is provided in the (first part of the) optimal solution of this LP in order to select the action to be performed at that particular decision state.

In the rest of this section we provide a detailed implementation of this computational scheme that leverages, $\mathcal{N}^{(tc)}$, the CRL-modeling tc-PN model of Section 5.2.2, in order to define the corresponding LP formulation.

5.3.1 The employed LP formulation

In Section 5.2.2, we provided a complete characterization of (i) the maximal steady-state throughput, f^*, that can be attained by the fluidized version of the considered CRL that is encoded by the tc-PN $\mathcal{N}^{(tc)}$, and (ii) the set of markings $OSS(\mathcal{N}^{(tc)})$ that can support this steady-state operation. The LP formulation that constitutes the fluid relaxation to be employed in this section essentially tries to drive the net $\mathcal{N}^{(tc)}$ from the marking $\hat{\mathbf{m}}$ of GSPN $\mathcal{N}$ that corresponds to the current decision point in the considered CRL, to a marking $\tilde{\mathbf{m}} \in OSS(\mathcal{N}^{(tc)})$, while minimizing the experienced loss during this transition w.r.t. the maximal possible outflow that is defined by f^*. The resulting optimal control problem belongs to the class of optimal control problems for the considered tc-PN models that has been investigated in Chapter 7 of [44]. Next we adapt the results of that work to the CRL-modeling tc-PN $\mathcal{N}^{(tc)}$ and the particular optimal control problem that is considered in this work.

As in [44], we shall derive the sought LP formulation in discretized time, where the time-discretizing (or "sampling") period will be set equal to some value Δt. Furthermore, for reasons that will become clear in the sequel, we stipulate that

$$\Delta t < \min_{j=1,\ldots,M} \left\{ \frac{1}{\mu(t_j)} \right\} \tag{5.49}$$

The above discretization of time induces a *discrete-time controlled continuous PN*, $\mathcal{N}^{(dt)}$, from the original tc-PN model $\mathcal{N}^{(tc)}$. Letting $\mathbf{m}(k)$ denote the marking of net $\mathcal{N}^{(dt)}$ at period k, the one-time-step transitional dynamics of this net satisfy the following state equation

$$\mathbf{m}(k+1) = \mathbf{m}(k) + \Delta t \cdot \Theta \cdot \mathbf{w}(k) \tag{5.50}$$

In Equation 5.50, Θ denotes the flow matrix of the underlying net $\mathcal{N}$, and $\mathbf{w}(k)$ denotes the instantaneous firing levels for the various transitions $t \in T$, that are presumed to be kept constant during the considered time interval Δt.

Furthermore, for a well-defined operation of the net $\mathcal{N}^{(dt)}$, the input variable $\mathbf{w}(k)$ that drives the dynamics of Equation 5.50 must satisfy the following additional constraints:

$$\forall j = 1, \ldots, M, \;\; \mathbf{w}(t_j; k) \leq \mu(t_j) \cdot \mathbf{m}(p_{jp}; k) \tag{5.51}$$

$$\mathbf{w}(k) \geq \mathbf{0} \tag{5.52}$$

$$\forall j = 1, \ldots, M, \;\; \mathbf{m}(p_{jw}; k+1) \geq 0 \;\; ; \;\; \mathbf{m}(p_{jb}; k+1) \geq 0 \tag{5.53}$$

Then, Proposition 7.6 of [44] implies the following properties for the dynamics of the dt-PN $\mathcal{N}^{(dt)}$:

Proposition 10 *Consider the dt-PN $\mathcal{N}^{(dt)}$ that is induced from the CRL-modeling ct-PN $\mathcal{N}^{(ct)}$ under time discretization with a sampling period Δt that satisfies the condition of Equation 5.49. Furthermore, suppose that the one-time-step transitional dynamics of net $\mathcal{N}^{(dt)}$ satisfy Equations 5.50–5.53. Then, net $\mathcal{N}^{(dt)}$ possesses the following properties:*

1. *All the markings $\mathbf{m}(k), \; k = 1, 2, \ldots,$ that are reached by net $\mathcal{N}^{(dt)}$, when initialized at any initial marking $\mathbf{m}_0 \geq \mathbf{0}$, are nonnegative.*

2. $\forall p_{jp} \in P, \; j = 1, \ldots, M, \; \mathbf{m}_0(p_{jp}) > 0 \Longrightarrow \mathbf{m}(p_{jp}; k) > 0, \; \forall k = 1, 2, \ldots.$

Part 1 of Proposition 10 guarantees that, under the condition of Equation 5.49, the dt-PN $\mathcal{N}^{(dt)}$ is a valid approximation of the dynamics of the ct-PN $\mathcal{N}^{(ct)}$ w.r.t. the preservation of the nonnegativity of the net marking.[6] On the other hand, part 2 of Proposition 10 replicates in the context of the dt-PN $\mathcal{N}^{(dt)}$, the fact that places p_{jp}, that are the input places for the timed transitions in net $\mathcal{N}^{(ct)}$, can be emptied only in the limit. These results are

[6] This remark pertains especially to the places $p_{jp}, \; j = 1, \ldots, M$, since, for places p_{jw} and p_{jb}, nonnegativity is enforced explicitly by the constraint of Equation 5.53. Also, the correct marking of resource and monitor places is guaranteed by the corresponding p-semiflows of the net.

complemented, and further strengthened, by the following proposition, that constitutes an adaptation to the considered net $\mathcal{N}^{(dt)}$ of the result that appears in Theorem 7.9 of [44].

Proposition 11 *A marking* $\mathbf{m}'$ *is reachable in net* $\mathcal{N}^{(dt)}$ *iff it is reachable in the untimed dynamics of net* $\mathcal{N}^{(tc)}$ *with a sequence that never empties an already marked place.*

It can be easily checked that the transition sequence σ that was employed in the proof of Proposition 9, satisfies the condition of Proposition 11, and therefore, Proposition 11 enables the extension of the reachability result of Proposition 9 to the operational context of net $\mathcal{N}^{(dt)}$; i.e., starting from any initial marking $\mathbf{m}$ of net $\mathcal{N}^{(dt)}$, we can drive this net to the marking set $OSS(\mathcal{N}^{(tc)})$ in a *finite* number of periods Δt. This realization subsequently leads to the following LP formulation for the fluid relaxation of the original scheduling problem that is pursued in this section:

$$\min \sum_{k=0}^{H} (f^* - \mathbf{w}(t_M; k)) \tag{5.54}$$

s.t.

$$\forall k = 0, 1, \ldots, H, \;\; \mathbf{m}(k+1) = \mathbf{m}(k) + \Delta t \cdot \Theta \cdot \mathbf{w}(k) \tag{5.55}$$

$$\forall k = 0, 1, \ldots, H, \; \forall j = 1, \ldots, M, \;\; \mathbf{w}(t_j; k) \leq \mu(t_j) \cdot \mathbf{m}(p_{jp}; k) \tag{5.56}$$

$$\forall k = 0, 1, \ldots, H, \; \mathbf{w}(k) \geq \mathbf{0} \tag{5.57}$$

$$\forall k = 0, 1, \ldots, H, \; \forall j = 1, \ldots, M, \;\; \mathbf{m}(p_{jw}; k+1) \geq 0 \;\; ; \;\; \mathbf{m}(p_{jb}; k+1) \geq 0 \tag{5.58}$$

$$\mathbf{m}(0) = \hat{\mathbf{m}} \tag{5.59}$$

The LP of Equations 5.54–5.59 is formulated over a finite time-horizon $H+1$ that is selected a priori as one of the problem parameters. During this time-horizon, the considered LP tries to determine the control variables $\mathbf{w}(k),\ k = 0, \ldots, H$, for the underlying dt-PN $\mathcal{N}^{(dt)}$ so that the total amount of fluid output by this net over the considered time-horizon is maximized. But for a sufficiently long time-horizon $H + 1$, this objective will be equivalently attained by trying to drive the net $\mathcal{N}^{(dt)}$ from its current marking $\hat{\mathbf{m}}$ to a marking $\tilde{\mathbf{m}} \in OSS(\mathcal{N}^{(tc)})$, while minimizing the loss experienced during this transition w.r.t. the total fluid that would be output by net $\mathcal{N}^{(dt)}$ if it was operated at the maximal flow rate f^* that was determined in Section 5.2.2. The LP objective that is stated in Equation 5.54 adopts this last perspective.[7]

On the other hand, the constraints of Equations 5.55–5.58 essentially replicate the constraints of Equations 5.50–5.53 on the pricing of the variables $\mathbf{w}(k)$, for every period $k = 0, 1, \ldots, H$. Collectively, they guarantee the proper functioning of the net $\mathcal{N}^{(dt)}$, and the satisfaction of the properties of Propositions 10 and 11 by the resulting operation.

Finally, the constraint of Equation 5.59 simply sets the marking $\hat{\mathbf{m}}$, corresponding to the current decision point of the underlying CRL, as the initial marking of net $\mathcal{N}^{(dt)}$.

5.3.2 The induced scheduling policy

After we have formulated and solved the LP relaxation of Equations 5.54–5.59 at some decision state $\hat{\mathbf{m}}$, the next step is to interpret the obtained optimal solution for this LP to a scheduling policy for the underlying CRL. The subsequent discussion essentially replicates the corresponding logic of Section 4.1 in the semantics of the CRL-modeling GSPN $\mathcal{N}$. In particular, in this new modeling framework, we want to use the optimal solution of the LP of Equations 5.54–5.59 as a "guide" in the selection of the next tangible marking $\mathbf{m}$ for the

[7] The provided explanation for the LP objective of Equation 5.54 also suggests the following criterion for assessing the adequacy of the length $H + 1$ for the time-horizon that is employed by the considered LP: The optimal value for this LP should be invariant to any extensions of the selected time-horizon $H + 1$ by one or more extra periods. It is this realization that has driven our preference for this particular representation of the LP objective.

CRL-modeling GSPN $\mathcal{N}$ among the set of tangible markings that is defined by the tangible reach $\mathcal{TR}(\hat{\mathbf{m}})$.

To effect this selection, let us denote by $\mathbf{m}^*(1)$ the marking in the obtained optimal solution for the LP relaxation for $k = 1$. Then, the proposed scheduling policy will select the next tangible marking $\tilde{\mathbf{m}} \in \mathcal{TR}(\hat{\mathbf{m}})$ for the CRL-modeling GSPN $\mathcal{N}$, through the following rule:

$$\tilde{\mathbf{m}} \in \arg \min_{\mathbf{m} \in \mathcal{TR}(\hat{\mathbf{m}})} \sum_{j=1,\ldots,M} |\mathbf{m}(p_j) - \mathbf{m}^*(p_j; 1)| \tag{5.60}$$

In more natural terms, the criterion of Equation 5.60 seeks to select a tangible marking $\mathbf{m} \in \mathcal{TR}(\hat{\mathbf{m}})$ that matches the marking $\mathbf{m}^*(1)$ at the places p_{jp} that enable the timed transitions $t_j,\ j = 1, \ldots, M$, as much as possible (w.r.t. the employed l_1-norm).

Furthermore, a secondary criterion that we have used to break any ties that are generated through the criterion of Equation 5.60, is as follows:

$$\tilde{\mathbf{m}} \in \arg \min_{\mathbf{m} \in \mathcal{TR}(\hat{\mathbf{m}})} \sum_{\forall p_\in \{p_{jx}: j=1,\ldots,M; x=w,p,b\}} |\mathbf{m}(p) - \mathbf{m}^*(p; 1)| \tag{5.61}$$

This new criterion selects a tangible marking $\mathbf{m} \in \mathcal{TR}(\hat{\mathbf{m}})$ that has the smallest l_1-distance from the marking $\mathbf{m}^*(1)$ w.r.t. the sub-marking that is defined by the places p of net $\mathcal{N}$ that model the processing stages of the underlying CRL.

5.4 Extending the presented methodology to other RAS classes

In this section we briefly discuss the extension of the FR-based scheduling method for the CRL throughput maximization problem that has been developed in this work, to the scheduling problem of maximizing the throughput of more complex RAS. In particular, we focus on the class of Disjunctive–Conjunctive (DC–) RAS, that has been studied extensively in [59].

From a modeling standpoint, the class of DC-RAS supports the concurrent execution of a number of process types. Furthermore, for each such process type, this new class

allows for (i) more arbitrary resource allocation requests by the corresponding processing stages than the CRL model considered in this work, and (ii) routing flexibility (i.e., an instance of these process types can execute through more than one sequence of processing stages). The monograph [59] provides (a) a detailed characterization of the structure and the operation of these RAS by means of the PN modeling framework, and (b) extensive methodology for the synthesis of efficient deadlock avoidance policies that take the form of linear inequalities on the net marking, and can be implemented through "monitor" places superimposed on the RAS-modeling PN.

A first complication for any attempted extension of the considered scheduling problem to the DC-RAS context arises from the fact that the notion of throughput maximization itself is ill-defined, since there is more than one process type. A reasonable way to circumvent this complication is by assuming that the production rates of all these process types must observe some predefined ratios; then, it is possible to maximize the total production rate, across all process types, by maximizing the production rate of any one of them. Furthermore, the work of [27] discusses how to encode these production-ratio requirements in the underlying PN model, while preserving all the corresponding theory of deadlock avoidance for these nets.

A second complication for the extension of the results that were developed in the previous parts of this chapter to DC-RAS, even when they are operated under the production-ratio constraints that were mentioned in the previous paragraph, arises from the presence of routing flexibility for the supported process types. When viewed in the light of the technical developments that were pursued in the earlier parts of this document, this routing flexibility implies that the underlying GSPN $\mathcal{N}$ will not possess the mono-t-semiflow property. This fact, in turn, requires the redefinition of the steady-state regime for the fluidized version of net $\mathcal{N}$ so that it allows the potential "shut down" of certain parts of this net, in particular, those routes of the different process types that might not be competitive. Once this new convention has been established, the computation of a maximizing flow vector $\mathbf{f}^*$ for the

controlled fluidized net $\mathcal{N}^{(tc)}$ can be attained through an LP formulation that is similar, in terms of its informational content, to the LP of Equations 5.20–5.23.[8]

A last point that needs some further discussion regarding the proposed extension of our main results to the DC-RAS model, concerns the second part of Proposition 10. We remind the reader that this part implies that in the operation of the discrete-time PN model $\mathcal{N}^{(dt)}$ that is induced from the fluidized net $\mathcal{N}^{(tc)}$, a marked place that feeds a timed transition of the net will never get empty. This could be a potential complication in view of the aforementioned need to "shut down" certain parts of the net in the steady-state markings that support its optimized operation.

But this issue is immediately resolved in any real-time implementation of the proposed method that starts the underlying RAS in its empty state, and consistently guides it through those markings that are competitive markings according to the selection logic of Equations 5.60–5.61. Such an operational scheme will never route any process instances in the direction of those processing stages that are not competitive according to the flow-maximizing LP of Equations 5.20–5.23, and therefore, the underlying network $\mathcal{N}$ will never mark any places that will have to be emptied by the considered LP relaxation.

Based on all the above discussion, it should be clear that the FR-based scheduling method that has been developed in this work, is effectively extensible to the broader class of DC-RAS. This discussion also reveals the structure, as well as the modeling and analytical capabilities, that are attained when the FR-based scheduling methodology is pursued through the PN-based modeling framework, according to the lines that were specified in this chapter.

5.5 Limitations of the FR-based scheduling method

In this section, we take advantage of the increased expressive power of the PN modeling framework for the considered scheduling problem that has been developed in this chapter,

[8] We emphasize, however, that this claim presumes that the structure of the DC-RAS modeling PN $\mathcal{N}$ will also encode, both, the applied deadlock avoidance policy and the imposed production-ratio constraints.

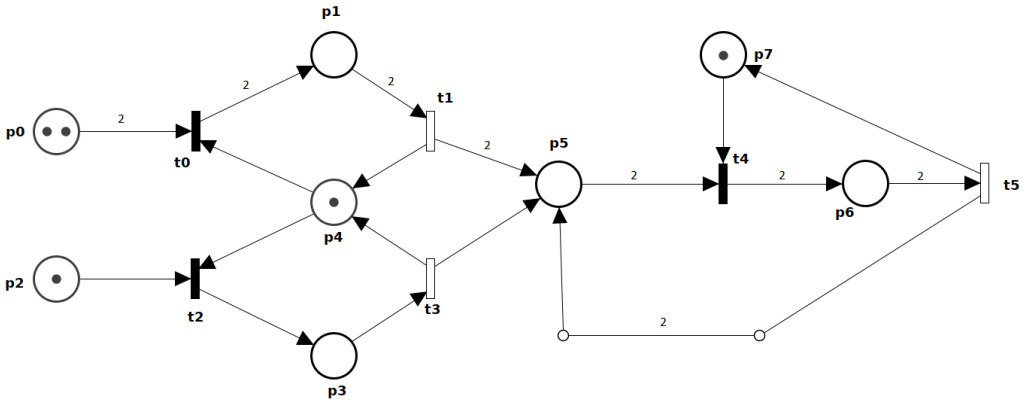

Figure 5.3: The GSPN model studied in Section 5.5.1

in order to identify some more concrete reasons that might be behind the suboptimality of the FR-based scheduling policy. Along these lines, we present two particular cases. In the first case, the FR-based scheduling method fails to make an optimal decision due to its inability to discern some discrete elements in the material-flow dynamics of the original scheduling problem that are invisible to the relaxing LP formulation. In the second case, the FR-based scheduling method fails to properly control some starvation effects in the dynamics of the underlying GSPN because, once again, these effects are not experienced in the relaxed, fluid dynamics. Furthermore, the insights that are obtained from this second case, also suggest a revision for (part of) the "action selection" logic of Section 5.3.2.

5.5.1 Limitations due to some quantization in the material-flow dynamics of the original GSPN model that is not visible to the relaxing LP

In this subsection, we consider the GSPN model depicted in Figure 5.3, where the solid-black transitions t_0, t_2 and t_4 constitute untimed transitions, while the remaining transitions t_1, t_3 and t_5, which are depicted as white bars, are timed. The firing rates for the timed transitions are $\lambda_i,\ i = 1, 3, 5$. Furthermore, a unit reward is associated with the firing of the timed transition t_5.

A simple inspection of the depicted PN reveals that the single repetitive behavior of this net is given by the sequence $t_4t_5t_4t_5t_4t_5\ldots \equiv (t_4t_5)^+$, and this behavior results in a long-term throughput of $(1/\lambda_5)^{-1} = \lambda_5$. Furthermore, this behavior is activated as soon as place p_5 acquires two tokens. These two tokens can be obtained from places p_0 and/or p_2, through the respective firing of the transition sequences t_0t_1 and t_2t_3. Finally, the presence of place p_4, and its respective connectivity to transitions t_0, t_1, t_2 and t_3, imply that only one of these two transition sequences can be active at any point in time.

More specifically, the firing of transition sequence t_0t_1 has an expected duration of $1/\lambda_1$ time units, and brings two tokens in place p_5. On the other hand, the firing of the transition sequence t_2t_3 has a duration of $1/\lambda_3$ time units, and brings only one token in place p_5. Hence, in this second case, for the proper activation of the repetitive behavior of the net, it is necessary to additionally fire the transition sequence t_0t_1, and therefore, in this case, the repetitive behavior of the net will be activated after an expected total time of $(1/\lambda_1) + (1/\lambda_3)$. Clearly, a policy that seeks to maximize the long-term throughput of the considered net when started from the marking depicted in Figure 5.3, must fire transition t_0 at this marking.

Next, let us consider what is the decision that is reached when applying the FR-based scheduling policy of the previous sections in the depicted situation.

Recognizing that the target throughput in the considered case is equal to λ_5, as discussed above, the corresponding relaxing LP of Equations 5.54–5.59 takes the following form for the considered case:

$$\min_{\mathbf{w}\geq\mathbf{0};\mathbf{m}\geq\mathbf{0}} \quad \sum_{k=0}^{H} \big(\lambda_5 - \mathbf{w}(t_5;k)\big)$$

s.t.

$$\begin{aligned}
\mathbf{m}(k+1) &= \mathbf{m}(k) + \Delta t \cdot \Theta \cdot \mathbf{w}(k), \qquad k = 0,\ldots,H \\
\mathbf{w}(t_i;k) &\leq \lambda_i \cdot \frac{\mathbf{m}(p_j;k)}{1 + I_{t_i\in\{t_1,t_5\}}}, \qquad \forall t_i \in \{t_1,t_3,t_5\},\ \forall p_j \in^\bullet t_i,\ k = 0,\ldots,H \\
\mathbf{w}(t_5;H) &= \lambda_5
\end{aligned}$$

In the above LP formulation, Θ is the flow matrix of the PN that is depicted in Figure 5.3, and Δt is a scalar such that $\Delta t < \min\{1/\lambda_1, 1/\lambda_3, 1/\lambda_5\}$. By solving this LP, and applying the selection logic of Section 5.3.2, the selected transition to be fired at the depicted marking is:

- t_0 if $2\lambda_1 > \lambda_3$;
- t_2 if $2\lambda_1 < \lambda_3$;
- any of t_0 and t_2 if $2\lambda_1 = \lambda_3$.

In order to understand the rationale of this selection scheme, the reader should notice the following: (i) Upon the firing of untimed transition t_0, transition t_1 gets an enabling degree of 1.0, and the same happens to transition t_3 when untimed transition t_2 fires. (ii) On the other hand, the outflow of transition t_1 towards place p_5 during this activation, is twice the outflow of transition t_3 towards the same place. Hence, the FR-based scheduling policy tries to maximize the total inflow to place p_5 in the early periods of the planning horizon of the relaxing LP, since this inflow to place p_5 will enable to the maximum possible degree the repetitive behavior $(t_4 t_5)^+$ and the collection of the targeted reward through the firing of the corresponding transition t_5. But as we discussed in the opening part of this subsection, this rationale does not capture the operational reality of the underlying GSPN, where the activation of the repetitive behavior $(t_4 t_5)$ will take place only when there are two tokens in place p_5.

As a more positive remark, we also notice that the limitation of the FR-based scheduling policy that was described in this example, will not be encountered in the context of the CRL – and even the DC-RAS – throughput maximization problem that has been considered in this work. In both of these cases, all the arcs that connect the timed transitions of the corresponding GSPNs with the input "process" places of these transitions, have a unitary weight, a fact that reflects the atomic nature of the processed entities in these systems. Hence, the asymmetry in the dynamics of the firing sequences $t_0 t_1$ and $t_2 t_3$ that was

experienced in the case of the the GSPN of Figure 5.3, cannot take place in the GSPNs that model the aforementioned RAS.

5.5.2 Limitations due to starvation effects that do not appear in the fluid dynamics of the LP relaxation

In this subsection, we will consider CRLs consisting of two workstations, WS_1 and WS_2, with buffer sizes B_1 and B_2, and supporting a process plan with three processing stages $J_j,\ j = 1, 2, 3$, such that $W(J_1) = W(J_3) = WS_1$ and $W(J_2) = WS_2$. Furthermore, we assume that the processing times for each of the three processing stages are exponentially distributed, with corresponding instantaneous rates $\mu_j = 1,\ j = 1, 2, 3$. And we also remind the reader that for the considered CRLs, the maximally permissive DAP takes the form

$$\hat{\mathbf{s}}_1 + \hat{\mathbf{s}}_2 \leq B_1 + B_2 - 1$$

where $\hat{\mathbf{s}}_i,\ i = 1, 2, 3$, denotes the number of process instances located in workstation $WS(J_i)$ in order to execute the corresponding processing stage J_i; more specifically, the process instances that are included in $\hat{\mathbf{s}}_i$ might wait for the execution of the processing stage J_i, are currently executing this processing stage, or have completed the execution of this processing stage but have not advanced yet to the next required workstation.

In the following, we shall focus on some vanishing states that are also decision states, and have the following form:

$$\hat{\mathbf{s}} = (0, B_1 - 2, 0, 1, B_2 - 1, 1, 0)$$

Given the CRL state semantics that were introduced in Section 2.3.2, the above form implies that, in the considered states $\hat{\mathbf{s}}$, there are: $(B_1 - 2)$ process instances in the "after-processing" status of processing stage J_1; $(B_2 - 1)$ process instances in the "after-processing" status of processing stage J_2; 1 process instance in the "before-processing" status of pro-

cessing stage J_3; and, finally, 1 process instance is being processed in processing stage J_2.

On the other hand, the states in the tangible reach $\mathcal{TR}(\hat{\mathbf{s}})$ are distinguished into two classes: (i) class $\mathcal{C}_1$ containing all those tangible states that allocate the currently idle server of workstation WS_1 to a process instance executing processing stage J_1; and (ii) class $\mathcal{C}_2$ containing those tangible states where the server of workstation WS_1 is allocated to a process instance executing processing stage J_3. In the following, we shall further assume that the timed dynamics of the considered CRL are such that the optimal decision at state $\hat{\mathbf{s}}$ belongs in class $\mathcal{C}_1$.

It is not hard to see that the aforementioned state class $\mathcal{C}_1$ consists of all those states with the form

$$\tilde{\mathbf{s}}(i) = (1, B_1 - 2 - i, i, 1, B_2 - 1 - i, 1 + i, 0), \quad i = 0, ..., \min(B_1 - 2, B_2 - 1)$$

More specifically, state $\tilde{\mathbf{s}}(0)$ is obtained by simply loading a new process instance in workstation WS_1 and initiating its first processing stage, without advancing any of those process instances that are already in the system. State $\tilde{\mathbf{s}}(1)$ is obtained by (a) first advancing a completed process instance in workstation WS_2 to workstation WS_3, (b) subsequently advancing a process instance that has completed the processing of its first processing stage at workstation WS_1 to workstation WS_2, and, finally, (c) loading a new process instance to workstation WS_1 and initiating the processing of the first stage of this process instance. The remaining states $\tilde{\mathbf{s}}(i),\ i \geq 2$, are obtained by iterating accordingly the above process-advancing scheme for the already loaded process instances, before loading the new process instance to workstation WS_1.

Furthermore, numerical experimentation with the aforementioned structures for values of $B_1 = 3, \ldots, 10$ and $B_2 = 3, \ldots, 10$ has shown that, when class $\mathcal{C}_1$ contains the optimal

decision for state $\hat{\mathbf{s}}$, this optimal decision is the state

$$\tilde{\mathbf{s}}(i) = (1, B_1 - 2 - i, i, 1, B_2 - 1 - i, 1 + i, 0) \text{ for } i = \min(B_1 - 2, B_2 - 1)$$

An intuitive explanation of this finding can be provided as follows: First, it is easy to check that each of the states $\tilde{\mathbf{s}}(i),\ i = 0, 1, \ldots, \min(B_1 - 2, B_2 - 1)$, has the buffers of both workstations WS_1 and WS_2 fully allocated. Furthermore, this regime will persist as long as the server of workstation WS_1 remains preoccupied with the processing of the currently allocated process instance (executing processing stage J_1). Hence, during this time interval, workstation WS_2 can work only on the i process instances that were advanced to it during the transition from the decision state $\hat{\mathbf{s}}$ to the corresponding state $\tilde{\mathbf{s}}(i)$. The aforementioned optimal decision seeks to maximize the available "work-in-process (WIP)" inventory for the server of workstation WS_2.

However, when we applied the FR-based scheduling method to some instantiations of the decision state $\hat{\mathbf{s}}$ that is considered in this discussion, the returned decision was represented by the tangible state $\tilde{\mathbf{s}}(1) = (1, B_1 - 3, 1, 1, B_2 - 2, 2, 0)$, which, according to the above discussion, is suboptimal. This is happening because the "fluid" model of the considered CRLs does not face the problem of blocking discussed in the previous paragraph, and therefore, the optimal solution of the LP relaxation does not have any incentive to allocate the buffers appropriately in order to minimize the starvation probability for the server of workstation WS_2. This misallocation of the buffering capacity by the optimal solution of the relaxing LP eventually is passed in the scheduling policy that is derived for the underlying CRL through the secondary selection rule of Section 5.3.2, since all the states $\tilde{\mathbf{s}}(i) \in \mathcal{C}_1$ correspond to the same server allocation for workstation WS_1, and therefore, they are equivalent with respect to the primary selection rule of that section.

Along similar lines to the above remarks, we have also noticed that, for a CRL that has no deadlock-free unsafe states and is operated under the optimal linear DAP ($A\mathbf{s} \leq b$),

when two tangible states $\mathbf{s_1}$ and $\mathbf{s_2}$ are in the tangible reach of a decision state $\mathbf{s}$, and it further holds that

- $\mathbf{s}_1$ and $\mathbf{s}_2$ have the same server allocation;
- $\mathbf{s}_1$ and $\mathbf{s}_2$ have the same number of parts at each workstation; and
- $A\mathbf{s}_1 \leq A\mathbf{s}_2$,

then, for the considered problem of the throughput maximization of this line, the optimal relative value of state $\mathbf{s}_1$ is no less than the optimal relative value of state $\mathbf{s}_2$; i.e., state $\mathbf{s}_2$ does not correspond to an optimal decision. We have not pursued a formal proof for this last remark, but it also true that, in all of our experiments, we have not found a single example that would counter this claim. On the other hand, the FR-based scheduling policy that is based on the selection logic of Section 5.3.2, might fail to recognize this dominance relationship among the aforementioned states $\mathbf{s}_1$ and $\mathbf{s}_2$.

Closing the discussion of this subsection, we want also to notice that, besides the interesting conceptual insights and some potential new research themes that are defined by this discussion, an important practical implication of this discussion is the realization that the secondary selection rule of Section 5.3.2 might not be very pertinent after all. Indeed, all our experimental findings indicate that the fluid-relaxation models that have been developed in this work, provide excellent guidance for the allocation of the system servers (which admittedly is the most major concern in the considered scheduling problems); but the attempt to resolve ties among competing states with the same server allocation on the basis of the information on buffer allocation that is provided by the optimal solution of the relaxing LP, might end up being misguided. More work is needed in the resolution of this last issue, and in the meantime, breaking the corresponding ties completely randomly might be a more pertinent decision mechanism.[9] Motivated by this last remark, in the next chapter

[9]This statement would be especially true for the example that was provided in the first part of this subsection, since such a randomizing mechanism would give a much larger probability to states $\tilde{\mathbf{s}}(i)$ with larger values of i than the current selection logic of Section 5.3.2.

we also consider possible corrective mechanisms that can detect and correct decisions of the FR-based scheduling policy that might end up being suboptimal.

CHAPTER 6

PERFORMANCE ENHANCEMENT OF THE FR-BASED SCHEDULING POLICY

The FR-based scheduling policy that was presented in Chapters 4 and 5 has been shown to outperform significantly any other practical policies that can be contemplated for the target RAS. It also gives near-optimal results when assessed against the optimal scheduling policies that are specified by the MDP formulation of the corresponding scheduling problem. Nevertheless, this policy remains a *heuristic* solution to the considered scheduling problem, and, in fact, the last section of Chapter 5 identified certain conditions under which the policy decisions might be expected to be suboptimal. Hence, in this chapter we develop some additional tools that establish a capability to detect such suboptimal decisions by the FR-based scheduling policy and correct them.

The presented tools can be applied locally at the different decision points of the underlying MDP, and the selection of these decision points can be performed in such a way that any attained local improvements can have the most extensive impact on the performance of the overall scheduling policy. Furthermore, the attempted improvements can be performed either (i) in an "off-line" mode that uses simulation in order to identify and assess potential modifications of the current scheduling policy that might lead to an enhanced performance for the underlying RAS, or (ii) in an "on-line" mode that uses information gathered during the real-time operation of the system under the current scheduling policy, in order to identify potential improvements for this policy and assess their expected impact on the system performance.

From a methodological standpoint, the presented results are based on (i) some fundamental results that are borrowed from the area of the sensitivity analysis of Markov reward processes [8], and (ii) the central role of the notion of the *"state potential"* in this anal-

ysis.[1] On the other hand, the computational tractability of the presented developments is established through statistical methods that enable (iii) the evaluation of the system performance and the aforementioned potentials through a sample-path based computation, and (iv) a systematic and robust comparison of the obtained estimates of these quantities. In particular, the task of selecting reliably the option that is expected to have the best performance, according to the obtained estimates of the employed performance indices, across all the entertained options, is based on a body of results from statistical inference that are collectively known as *"ranking & selection (r&s)"* [31].

We should also notice, at this point, that there have been some prior attempts to develop "corrective methods" to scheduling policies that have been obtained through a relaxing approach to an original MDP formulation. Perhaps the most prominent among these approaches in the current literature is that presented in [46]. This approach has been developed for multi-class queueing networks under a typical assumption of infinite-capacity buffers, and tries to determine an efficient scheduling policy that will drain these networks from a large accumulated backlog in the network buffers. To this end, the method of [46] first formulates and solves "off-line" a (deterministic) optimal control problem based on a fluidized version of the underlying system dynamics, and subsequently it tries to guarantee the feasibility and the efficiency of the computed scheduling policy by trying to keep the system operation away from a "boundary region" that is defined by the drainage of the system buffers; the tool for attaining this last objective in [46] is the notion of "safety stock". As remarked, however, in [25], the estimation and preservation of the right amounts of "safety stock" is a challenging problem, and, furthermore, the amounts of the required safety stocks can be quite extensive. In addition, the notion of "safety stock" can be further challenged by the presence of finite buffer capacities. Besides rendering impossible the preservation of some large amounts of safety stock that might be requested by the aforementioned method, the presence of finite buffering capacities in the underlying scheduling

[1] State potentials are also known as the *"relative value function"* in the relevant MDP literature.

problem defines an additional "boundary' region" that must be avoided during the real-time execution of the pre-determined scheduling policy, which is defined by a complete allocation of the limited buffers. The avoidance of this new "boundary region" requires the preservation of a "safety stock" of free buffering capacity, and the simultaneous attendance of both "safety stock" requirements might result in an over-constrained problem.

On the other hand, our FR-based scheduling policy tries to account for the impact of these "boundary" effects on the system performance by (re-)formulating and solving the employed LP relaxation at each decision point. In the same spirit, the eventually sought improvements of the original scheduling policy will be determined at the decision-point level; i.e., they constitute isolated revisions of the actions to be selected by the applied policy at a targeted set of decision points. In fact, when viewed from this standpoint, the policy-improving methodology that is pursued in this chapter, bears stronger similarity to the ideas and the techniques that underlie the "policy iteration (PI)" method for the solution of infinite-horizon, average-reward MDPs [57, 8], especially those implementations of this method that seek to evaluate the performance of the running policy at each iteration and solve the corresponding Poisson equation through simulation [6, 8, 14].

Finally, closing this brief literature review on "correcting" methods for FR-based scheduling policies, we should also mention the approach presented in [7]. Actually, this approach does not constitute a "correcting" method for a pre-determined FR-based scheduling policy in the strict sense, but it introduces a notion of "robustness" in the formulation of the relaxing LP that is solved at each decision point, in an attempt to account more explicitly for the statistical characteristics that determine the notion of randomness in the timing distributions of the underlying stochastic network. The experimental results that are reported in [7], indicate that the modified LP relaxation, based on the aforementioned notion of "robustness", can result in an improved scheduling policy for the underlying stochastic network.

In view of the above positioning of the main results of this chapter, the rest of it

is organized as follows: Section 6.1 reviews the results from the sensitivity analysis of infinite-horizon AR-MDPs that are at the basis of the developments that are presented in this chapter, adapts these results to the context of the particular scheduling problem and the corresponding MDP that are addressed in this work, and motivates the two improving methods for the FR-scheduling policy of Chapters 4 and 5 that are implied by these results (i.e., the "off-line" and the "on-line" improving method that were discussed in the opening paragraphs of this chapter). On the other hand, Sections 6.2 and 6.3 present the implementational details of the presented methods. In particular, Section 6.2 discusses the sample-path-based methods that are necessary for the computation of the different estimates that are necessary for a systematic comparison of the different decisions that are available at the selected decision point(s), while Section 6.3 presents the "ranking & selection" method that is employed for this comparison. The chapter concludes with Section 6.4 that presents a series of results which demonstrate and assess the efficacy of the developed methodology, and briefly discusses some further implementational details for the orchestration of the presented developments into a full-fledged policy-improving method.

6.1 Some fundamental results from the sensitivity analysis of infinite-horizon AR-MDPs and their implications for the potential improvement of the FR-based scheduling policy

In this section we consider a Markov reward process that is induced by any given stationary deterministic policy $\pi \in \Pi$ for the MDP that models the CRL scheduling problem addressed in this work, and we present some results regarding the sensitivity analysis of this Markov reward process. These sensitivity results subsequently enable us to motivate and outline the policy-improving methods that are the main theme of this chapter.

An induced Markov reward process In order to define the Markov reward process that is the focus of this section, we start by considering the continuous-time Markov chain

(CTMC) $\mathcal{M}(\pi)$ that is defined by any given stationary deterministic policy π for the MDP of Section 2.4, as follows:

The state space of this CTMC is the set S_a^T of the tangible states of the underlying state space S that models the untimed dynamics of the considered CRL, that are also admissible by the applied DAP Δ. The transition rate $q_{ij}(\pi)$ for any pair of states $(\mathbf{s}_i, \mathbf{s}_j) \in S_a^T \times S_a^T$, under the considered policy π, can be computed as follows:

$$q_{ij}(\pi) = \sum_{e_\zeta^d \in \mathcal{E}(\mathbf{s}_i):\ \mathbf{s}_l = f(\mathbf{s}_i, e_\zeta^d)\ \wedge\ a(\mathbf{s}_l;\pi) = \mathbf{s}_j} \mu_\zeta \tag{6.1}$$

We remind the reader that, according to the notation that was introduced in Chapter 2, in the above equation, (i) e_ζ^d denotes the uncontrollable event corresponding to the completion of processing stage J_ζ, $\zeta = 1, \ldots, M$; (ii) $f(\cdot, \cdot)$ denotes the state transition function modeling the untimed dynamics of the considered CRL; (iii) state $\mathbf{s}_l$ constitutes a decision point for the MDP that models the scheduling problem of maximizing the throughput of the considered CRL; and (iv) $a(\mathbf{s}_l; \pi)$ denotes the decision (or action) that is selected by the deterministic policy π at the decision point $\mathbf{s}_l$; furthermore, (v) an action selected at decision state $\mathbf{s}_l$ is essentially a tangible state $\mathbf{s}'$ in the tangible reach $\mathcal{TR}(\mathbf{s}_l)$ of state $\mathbf{s}_l$. Hence, in plain terms, the transition rate $q_{ij}(\pi)$ is equal to the total occurrence rate in state $\mathbf{s}_i$ of all those timed events e_ζ^d that lead to a decision state $\mathbf{s}_l$ where the selected decision under policy π is the tangible state $\mathbf{s}_j$. The transitional dynamics that underlie this definition, are depicted schematically in Figure 6.1.

The CTMC $\mathcal{M}(\pi)$ is turned into a Markov reward process by associating the following reward rate $r(\mathbf{s}_i)$ with every state $\mathbf{s}_i$ of $\mathcal{M}(\pi)$:

$$r(\mathbf{s}_i) = \begin{cases} \mu_M, & \text{if } e_M^d \in \mathcal{E}(\mathbf{s}_i) \\ 0, & \text{o.w.} \end{cases} \tag{6.2}$$

In order to simplify the exposition of the subsequent developments, $\mathcal{M}(\pi)$ is also uni-

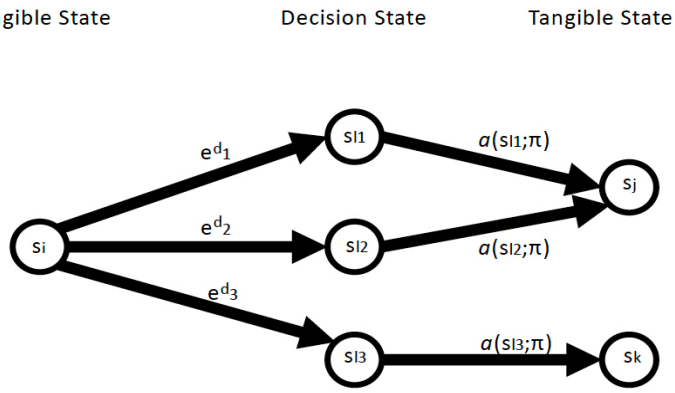

Figure 6.1: A schematic representation of the transitional dynamics that determine the transition rates $q_{ij}(\pi)$ of the CTMC $\mathcal{M}(\pi)$.

formized, with uniformizing rate $r_u = \sum_{i=1}^{L} \max_{j:W(J_j)=WS_i} \mu_j$, in order to obtain the discrete-time Markov chain (DTMC) $\hat{\mathcal{M}}(\pi)$. Letting also $\Delta t_u \equiv 1/r_u$, the one-step transition probability matrix $\hat{P}(\pi)$ for this last Markov chain is defined as follows:

$$\hat{P}(i,j;\pi) = \begin{cases} q_{ij}(\pi)\cdot\Delta t_u = \left(\sum_{e^d_\zeta \in \mathcal{E}(\mathbf{s}_i):\ \mathbf{s}_l = f(\mathbf{s}_i, e^d_\zeta)\ \wedge\ a(\mathbf{s}_l;\pi)=\mathbf{s}_j} \mu_\zeta\right)\cdot\Delta t_u, & \text{if } i \neq j \\ 1 - \sum_{j:j\neq i}\hat{P}(i,j;\pi) = 1 - \Delta t_u \cdot \sum_{j:j\neq i} q_{ij}(\pi), & \text{o.w.} \end{cases} \tag{6.3}$$

Finally, in the uniformized dynamics of the DTMC $\hat{\mathcal{M}}(\pi)$, the state rewards are given by the vector $\hat{\mathbf{r}}$, with components

$$\hat{\mathbf{r}}_i = r(\mathbf{s}_i)\cdot\Delta t_u = \begin{cases} \mu_M \cdot \Delta t_u, & \text{if } e^d_M \in \mathcal{E}(\mathbf{s}_i) \\ 0, & \text{o.w.} \end{cases} \tag{6.4}$$

The Poisson equation and performance derivatives for the Markov reward process $\hat{\mathcal{M}}(\pi)$ Let $\hat{\eta}(\pi)$ denote the long-term average reward of process $\hat{\mathcal{M}}(\pi)$ under policy π. Then, assuming that the considered policy π confines the long-term behavior of the underlying CRL in a single communicating class of the process state space S_a,[2] we have that

[2] In general, there is no straightforward guarantee that the arbitrary policies π that are considered in this chapter will induce a Markov chain $\mathcal{M}(\pi)$ with a single absorbing communicating class. In fact, we do not have similar formal guarantees even for the FR-based scheduling policy that was discussed in Chapters 4

$$\hat{\eta}(\pi) = \hat{\psi}(\pi)^T \cdot \hat{\mathbf{r}} \tag{6.5}$$

where $\hat{\psi}(\pi)$ is the column vector that represents the limiting distribution of Markov chain $\hat{\mathcal{M}}(\pi)$. Furthermore, it is well known that, under the same ergodicity assumptions, there exists a vector $\hat{\mathbf{g}}(\pi)$ that is defined up to an additive constant by the following equation [57, 8]:

$$\Big(I - \hat{P}(\pi)\Big) \cdot \hat{\mathbf{g}}(\pi) = \hat{\mathbf{r}} - \hat{\eta}(\pi)\mathbf{1} \tag{6.6}$$

Equation 6.6 is known as the *Poisson equation* in the corresponding literature. The matrix I that appears in the left-hand-side of this equation denotes an appropriately dimensioned identity matrix, while the vector $\mathbf{1}$ that appears in its right-hand-side is a vector with all its components being equal to one. Vector $\hat{\mathbf{g}}(\pi)$ is known as the *relative value function* for the underlying Markov reward process $\hat{\mathcal{M}}(\pi)$, and its components are also known as the corresponding *state potentials* (under the considered policy π).

Next, consider another stationary deterministic policy π' for the MDP of Section 2.4, with one-step transition probability matrix $\hat{P}(\pi')$ and limiting distribution $\hat{\psi}(\pi')$ for the corresponding DTMC $\hat{M}(\pi')$. From Equation 6.6 we have:

$$\begin{aligned}
\hat{\psi}(\pi')^T \cdot \Big((I - \hat{P}(\pi)\Big) \cdot \hat{\mathbf{g}}(\pi) = \hat{\psi}(\pi')^T \cdot \Big(\hat{\mathbf{r}} - \hat{\eta}(\pi)\mathbf{1}\Big) \quad &\Longrightarrow \\
\hat{\psi}(\pi')^T \cdot \Big(\hat{P}(\pi') - \hat{P}(\pi)\Big) \cdot \hat{\mathbf{g}}(\pi) = \hat{\psi}(\pi')^T \cdot \hat{\mathbf{r}} - \hat{\eta}(\pi)\Big(\hat{\psi}(\pi')^T \cdot \mathbf{1}\Big) \quad &\Longrightarrow \\
\hat{\psi}(\pi')^T \cdot \Big(\hat{P}(\pi') - \hat{P}(\pi)\Big) \cdot \hat{\mathbf{g}}(\pi) = \hat{\eta}(\pi') - \hat{\eta}(\pi) \quad &
\end{aligned} \tag{6.7}$$

In the above derivation we have used the additional facts that

$$\hat{\psi}(\pi')^T \cdot \hat{P}(\pi') = \hat{\psi}(\pi')^T \tag{6.8}$$

and 5. Although such a problem has not been detected in the numerical experiments that are reported in the various parts of this thesis, we notice that one way to regain the aforementioned ergodicity property for any policy π, is by considering a *randomized implementation* of this policy that, at each decision state, it implements the action-selection logic of policy π with a very high probability p, and selects completely randomly from the available actions at that decision point, with probability $1 - p$.

and

$$\hat{\psi}(\pi')^T \cdot \mathbf{1} = 1.0 \tag{6.9}$$

Equation 6.7 is a *"performance difference"* formula, since it characterizes the difference in the long-term performance of the underlying CRL as we switch from policy π to policy π'. We can see that this difference is determined by (i) the element-wise difference of the one-step transition probability matrices $\hat{P}(\pi)$ and $\hat{P}(\pi')$ for the DTMCs that are induced by these two policies, (ii) the relative value function $\hat{\mathbf{g}}(\pi)$ of policy π, and (iii) the limiting distribution $\hat{\psi}(\pi')$ of policy π'.

Furthermore, setting

$$\hat{P}_\delta(\pi, \pi') = (1 - \delta)\hat{P}(\pi) + \delta\hat{P}(\pi') \tag{6.10}$$

for some $\delta \in (0, 1)$, and applying the result of Equation 6.7 with respect to the policies that correspond to the matrices $\hat{P}(\pi)$ and $\hat{P}_\delta(\pi, \pi')$, we get

$$\hat{\psi}_\delta(\pi, \pi')^T \cdot \delta\Big(\hat{P}(\pi') - \hat{P}(\pi)\Big) \cdot \hat{\mathbf{g}}(\pi) = \hat{\eta}_\delta(\pi, \pi') - \hat{\eta}(\pi) \tag{6.11}$$

where $\hat{\psi}_\delta(\pi, \pi')$ and $\hat{\eta}_\delta(\pi, \pi')$ denote, respectively, the limiting distribution and the long-term throughput of the randomized policy that corresponds to the one-step transition probability matrix $\hat{P}_\delta(\pi, \pi')$. Furthermore, dividing both sides of Equation 6.11 by δ, and letting $\delta \to 0^+$, we get the equation

$$\hat{\psi}(\pi)^T \cdot \Big(\hat{P}(\pi') - \hat{P}(\pi)\Big) \cdot \hat{\mathbf{g}}(\pi) \;=\; \frac{d\hat{\eta}_\delta(\pi, \pi')}{d\delta}\bigg|_{\delta=0} \tag{6.12}$$

Equation 6.12 is a *"performance derivative"* formula. The quantity $\frac{d\hat{\eta}_\delta(\pi,\pi')}{d\delta}\Big|_{\delta=0}$ that appears in the right-hand-side of this equation, can be perceived as a *"directional derivative"* that characterizes the performance change for the underlying MDP as the applied

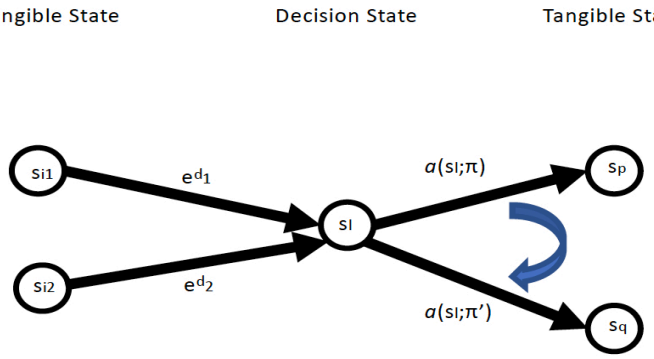

Figure 6.2: A schematic representation of the policy modifications that are considered in this chapter.

policy moves from the original policy π towards policy π' according to the randomizing scheme that is defined by Equation 6.10.

The reader should also notice that in the evaluation of the derivative $\left.\frac{d\hat{\eta}_\delta(\pi,\pi')}{d\delta}\right|_{\delta=0}$ according to the left-hand-side of Equation 6.12, both, the relative value function and the limiting distribution that are employed in this computation, are those corresponding to the originating policy π. This realization is important since it implies that in the sample-path-based computations that are pursued in the rest of this chapter, the application of the result of Equation 6.12 requires the observation of the underlying CRL under the operation of the currently applied policy π only; therefore, those computations that are based only on the result of Equation 6.12, can be performed in an "on-line" mode, during the "real-time" operation of the considered CRL.

Detecting policy-improving opportunities for the MDP of Section 2.4 In the rest of this section, we specialize the results of Equations 6.7 and 6.12 to the particular case where the deterministic policy π' is obtained from the currently applied deterministic policy π through the modification of a single decision, that is associated with some decision state $\mathbf{s}_l \in X$. The considered modification is depicted graphically in Figure 6.2. The characterization of the performed modification through Figure 6.2, when combined with the characterization of the one-step transition probability matrix $\hat{P}(\pi)$ through Equation 6.3,

further imply that, in the considered case, the matrix difference $\Delta\hat{P}(\pi,\pi') \equiv \hat{P}(\pi') - \hat{P}(\pi)$, that appears in Equations 6.7 and 6.12, will possess the following structure:

$$\Delta\hat{P}(i,k;\pi,\pi') = \begin{cases} -\mu_j \Delta t_u, & \text{if } \exists e_j^d \in \mathcal{E}(\mathbf{s}_i) \text{ s.t. } f(\mathbf{s}_i, e_j^d) = \mathbf{s}_l \ \wedge \ k = p \\ \mu_j \Delta t_u, & \text{if } \exists e_j^d \in \mathcal{E}(\mathbf{s}_i) \text{ s.t. } f(\mathbf{s}_i, e_j^d) = \mathbf{s}_l \ \wedge \ k = q \\ 0, & \text{o.w.} \end{cases} \tag{6.13}$$

Equation 6.13 subsequently implies the following forms for the performance-difference and the performance-derivative formulae of Equations 6.7 and 6.12:

$$\hat{\eta}(\pi') - \hat{\eta}(\pi) = \Delta t_u \Big(\sum_{i:\ \exists e_j^d \in \mathcal{E}(\mathbf{s}_i) \text{ s.t. } f(\mathbf{s}_i, e_j^d) = \mathbf{s}_l} \hat{\psi}(\mathbf{s}_i;\pi') \cdot \mu_j \Big) \Big[\hat{\mathbf{g}}(\mathbf{s}_q;\pi) - \hat{\mathbf{g}}(\mathbf{s}_p;\pi) \Big] \tag{6.14}$$

$$\left. \frac{d\hat{\eta}_\delta(\pi,\pi')}{d\delta} \right|_{\delta=0} = \Delta t_u \Big(\sum_{i:\ \exists e_j^d \in \mathcal{E}(\mathbf{s}_i) \text{ s.t. } f(\mathbf{s}_i, e_j^d) = \mathbf{s}_l} \hat{\psi}(\mathbf{s}_i;\pi) \cdot \mu_j \Big) \Big[\hat{\mathbf{g}}(\mathbf{s}_q;\pi) - \hat{\mathbf{g}}(\mathbf{s}_p;\pi) \Big] \tag{6.15}$$

Equations 6.14 and 6.15 imply that a policy modification of the type that is described in Figure 6.2 can result in an improvement of the performance of the underlying CRL only if

$$\hat{\mathbf{g}}(\mathbf{s}_q;\pi) \ > \ \hat{\mathbf{g}}(\mathbf{s}_p;\pi) \tag{6.16}$$

The magnitude of this improvement is determined by the difference $\hat{\mathbf{g}}(\mathbf{s}_q;\pi) - \hat{\mathbf{g}}(\mathbf{s}_p;\pi)$, and it is further modulated by the factors that multiply this difference in the two Equations 6.14 and 6.15. It can also be seen that, for Equation 6.15, this modulating factor is essentially the "steady-state" probability of visiting the considered decision state $\mathbf{s}_l$ under policy π, while, for Equation 6.14, the corresponding modulating factor is the "steady-state" probability of visiting the considered decision state $\mathbf{s}_l$ under policy π'.

The above remarks imply that in order to effect a performance improvement through a policy change of the type that is suggested by Figure 6.2, one needs to identify a decision

state $\mathbf{s}_l$ such that the current decision corresponds to a tangible state $\mathbf{s}_p \in \mathcal{TR}(\mathbf{s}_l)$ with $\hat{\mathbf{g}}(\mathbf{s}_p;\pi) < \max_{\mathbf{s}\in\mathcal{TR}(\mathbf{s}_l)} \hat{\mathbf{g}}(\mathbf{s};\pi)$. Furthermore, it makes sense to focus the search for a decision state $\mathbf{s}_l$ with the aforementioned property, to those decision states that are visited most frequently under the current policy π.

Finally, for the subsequent developments, it is also useful to notice that while the state potentials $\hat{\mathbf{g}}(\mathbf{s};\pi),\ \mathbf{s} \in S_a^T$, provide good guidance for identifying improving policy modifications of the type described in Figure 6.2, when one tries to assess the pertinence and the significance of such policy changes in an "off-line" mode, it is also possible to work more directly with estimates of the corresponding throughputs $\hat{\eta}(\pi)$ and $\hat{\eta}(\pi')$, obtained through simulation of the underlying CRL.

In the next section we address the issue of developing pertinent estimators for the throughput $\hat{\eta}(\pi)$ and the potentials $\hat{\mathbf{g}}(\mathbf{s};\pi),\ \mathbf{s} \in S_a^T$, for the considered Markov reward process $\hat{\mathcal{M}}(\pi)$, which is induced by some stationary deterministic policy π. Furthermore, in Section 6.3 we present a systematic methodology that will help us identify reliably improving modifications of the current policy π, based on some throughput and/or potential estimates that are obtained from the results of Section 6.2.

6.2 Sample-path-based estimation of the system throughput and of the state potentials under a given scheduling policy

In this section, we present some estimators for the throughput, $\hat{\eta}(\pi)$, and the potentials, $\hat{\mathbf{g}}(\mathbf{s};\pi),\ \mathbf{s} \in S_a^T$, that were introduced in the previous section. In the subsequent discussion it is assumed that the considered policy π induces a single absorbing communicating class for the DTMC $\hat{M}(\pi)$, and therefore, the throughput $\eta(\pi)$ and the potentials $\hat{\mathbf{g}}(\mathbf{s};\pi),\ \mathbf{s} \in S_a^T$, are well defined. Furthermore, from the overall discussion of the previous section, it should be clear that in regards to the potentials $\hat{\mathbf{g}}(\mathbf{s};\pi),\ \mathbf{s} \in S_a^T$, we are particularly interested in those states that belong in the aforementioned single communicating class, and therefore, they have a recurrent presence in the dynamics of the considered

DTMC $\hat{M}(\pi)$. The focus to this particular subclass of states subsequently enables us to leverage the regenerative nature of the dynamics of Markov chain $\hat{M}(\pi)$ within its absorbing communicating class.

From an organizational standpoint, the section consists of two major parts, with the first part addressing the estimation of the throughput $\hat{\eta}(\pi)$, and the second part addressing the estimation of the state potentials $\hat{\mathbf{g}}(\mathbf{s};\pi)$.

Estimating the throughput $\hat{\eta}(\pi)$ of the DTMC $\hat{M}(\pi)$ This part adapts to the considered problem of the estimation of the throughput $\hat{\eta}(\pi)$, some related results that are presented in [3]. Hence, let us consider a recurrent state $\mathbf{s}^*$ of the DTMC $\hat{M}(\pi)$, set $\mathbf{s}(0) = \mathbf{s}^*$ (i.e., initialize the process $\hat{M}(\pi)$ to state $\mathbf{s}^*$), and define the stopping time $\tau = \inf\{t > 0 : \mathbf{s}(t) = \mathbf{s}^*\}$ (i.e., τ is the "recurrence time" to state $\mathbf{s}^*$). Then, it is well known that

$$\hat{\eta}(\pi) = \frac{E\Big[\sum_{t=0}^{\tau-1} \hat{\mathbf{r}}\big(\mathbf{s}(t)\big)\Big]}{E[\tau]} \tag{6.17}$$

Equation 6.17 subsequently suggests the following estimator, $\widehat{\hat{\eta}(\pi)}$, for the throughput $\hat{\eta}(\pi)$: Simulate the DTMC $\hat{M}(\pi)$ initializing it to the selected state $\mathbf{s}^*$, and let τ_N denote the time of the N-th recurrence of the process to state $\mathbf{s}^*$. Then, set

$$\widehat{\hat{\eta}(\pi)} \;\equiv\; \frac{\sum_{t=0}^{\tau_N-1} \hat{\mathbf{r}}\big(\mathbf{s}(t)\big)}{\tau_N} \tag{6.18}$$

Estimator $\widehat{\hat{\eta}(\pi)}$ essentially employs the empirical means of the expectations $E[\tau]$ and $E\Big[\sum_{t=0}^{\tau-1} \hat{\mathbf{r}}\big(\mathbf{s}(t)\big)\Big]$ appearing in Equation 6.17, that are based on the N simulated recurrent cycles. Therefore, $\widehat{\hat{\eta}(\pi)}$ is strongly consistent (i.e., $\widehat{\hat{\eta}(\pi)} \to \hat{\eta}(\pi)$ as $N \to \infty$ w.p. 1), but it is also biased for any finite N. In [3] it is shown that the bias of $\widehat{\hat{\eta}(\pi)}$ is $O(1/N)$, and if necessary, it can be further reduced to $O(1/N^2)$ by applying certain techniques like "jack-knifing". Furthermore, the $O(1/N)$ dependence of the bias of $\widehat{\hat{\eta}(\pi)}$ on N implies that, when N takes fairly large values, this bias will be an order of magnitude smaller than

the st. deviation of this estimator, which behaves as $O(1/\sqrt{N})$; therefore, the existing bias in the above estimator $\widehat{\hat{\eta}(\pi)}$ is not expected to have a significant impact in the subsequent developments.

Estimating the state potentials $\hat{\mathrm{g}}(\mathrm{s};\pi)$ of the DTMC $\hat{M}(\pi)$ for its recurrent states s This part is based on corresponding developments that appear in [8, 14]. One way to motivate these developments is as follows: It is well known that a solution $\hat{\mathbf{g}}(\pi)$ for the Poisson equation of Equation 6.6, can be obtained by setting

$$\forall\, \mathbf{s}_i \in S_a^T,\;\; \hat{\mathbf{g}}(\mathbf{s}_i;\pi) = \lim_{L\to\infty} E\Big[\sum_{t=0}^{L-1}\Big(\hat{\mathbf{r}}\big(\mathbf{s}(t)\big) - \hat{\eta}(\pi)\Big)\;\Big|\;\mathbf{s}(0)=\mathbf{s}_i\Big] \tag{6.19}$$

Furthermore, from Equation 6.19 it follows that for $\mathbf{s}_j \neq \mathbf{s}_i$,

$$\gamma(\mathbf{s}_i,\mathbf{s}_j;\pi) \;\equiv\; \hat{\mathbf{g}}(\mathbf{s}_j;\pi) - \hat{\mathbf{g}}(\mathbf{s}_i;\pi) \;=\; E\Big[\sum_{t=0}^{\tau(i|j)-1}\Big(\hat{\mathbf{r}}\big(\mathbf{s}(t)\big) - \hat{\eta}(\pi)\Big)\;\Big|\;\mathbf{s}(0)=\mathbf{s}_j\Big] \tag{6.20}$$

where

$$\tau(i|j) = \inf\{t>0 : \mathbf{s}(t)=\mathbf{s}_i \mid \mathbf{s}(0)=\mathbf{s}_j\} \tag{6.21}$$

Finally, setting

$$\hat{\mathbf{g}}(\mathbf{s}^*;\pi) = 0 \tag{6.22}$$

for some arbitrary recurrent state $\mathbf{s}^*$, Equation 6.20 implies that the vector

$$\tilde{\mathbf{g}}(\mathbf{s}_i;\pi) = \begin{cases} 0, & \text{if } \mathbf{s}_i = \mathbf{s}^* \\ \gamma(\mathbf{s}^*,\mathbf{s}_i;\pi), & \text{o.w.} \end{cases} \tag{6.23}$$

is another valid solution for the Poisson equation of Equation 6.6.

Moreover, Equations 6.20 and 6.21 further imply that an estimator, $\widehat{\tilde{\mathbf{g}}(\mathbf{s}_i;\pi)}$, of $\tilde{\mathbf{g}}(\mathbf{s}_i;\pi)$, for some *recurrent* state $\mathbf{s}_i \neq \mathbf{s}^*$, can be obtained as follows: Consider a recurrent cycle of the DTMC $\hat{M}(\pi)$ with respect to the state $\mathbf{s}^*$, of length τ. If this recurrent cycle does not

visit state $\mathbf{s}_i$, then it cannot provide an estimate of $\tilde{\mathbf{g}}(\mathbf{s}_i;\pi)$. If, on the other hand, state $\mathbf{s}_i$ is visited during this recurrent cycle, then let τ_i denote the first period that state $\mathbf{s}_i$ is visited during this cycle. According to Equations 6.20 and 6.21, an estimate of $\tilde{\mathbf{g}}(\mathbf{s}_i;\pi)$ is provided by

$$\widehat{\tilde{\mathbf{g}}(\mathbf{s}_i;\pi)} = \sum_{t=\tau_i}^{\tau-1} \Big(\hat{\mathbf{r}}\big(\mathbf{s}(t)\big) - \widehat{\hat{\eta}(\pi)} \Big) \tag{6.24}$$

In Equation 6.24, $\widehat{\hat{\eta}(\pi)}$ is a previously obtained estimate of the throughput $\hat{\eta}(\pi)$; e.g., through the methodology that was discussed in the previous part of this section. Then, it can also be seen that the estimator $\widehat{\tilde{\mathbf{g}}(\mathbf{s}_i;\pi)}$ will be unbiased if the employed throughput estimate $\widehat{\hat{\eta}(\pi)}$ is unbiased; otherwise, it will be biased. Finally, a more robust estimate of $\tilde{\mathbf{g}}(\mathbf{s}_i;\pi)$ can be obtained by averaging the estimator of Equation 6.24 over N recurrent cycles with respect to state $\mathbf{s}^*$ that involve a visitation to the considered state $\mathbf{s}_i$, for some appropriately selected value of N.

6.3 A "ranking & selection" algorithm for identifying a performance-improving decision

As explained in Section 6.1, we are interested in improving modifications for any given policy π for the scheduling problem that is considered in this work, that adhere to the modification logic that is depicted in Figure 6.2. Furthermore, according to the discussion that was provided in the closing part of that section, such an improving modification can be detected by comparing either (i) the state potentials $\hat{\mathbf{g}}(\mathbf{s};\pi)$ of all states $\mathbf{s}$ that are in the tangible reach $\mathcal{TR}(\mathbf{s}_l)$ of the decision state $\mathbf{s}_l$ under consideration (when working in an "on line" mode), or (ii) the throughputs $\hat{\eta}(\pi')$ that are attained by the policies π' that result from the replacement of the current decision at the considered decision state $\mathbf{s}_l$ with another decision that corresponds to an alternative state $\mathbf{s} \in \mathcal{TR}(\mathbf{s}_l)$ (when working in an "off line" mode). In both cases, we are faced with the problem of selecting the state $\mathbf{s} \in \mathcal{TR}(\mathbf{s}_l)$ that maximizes the performance index of interest – i.e., the state potential $\hat{\mathbf{g}}(\mathbf{s};\pi)$, or the

throughput $\hat{\eta}(\pi')$ of the policies π' that result from the contemplated modifications – and we need to solve this problem while working with sample-path-based estimates of these quantities that can be obtained through the estimators that were defined in Section 6.2.

The problem of effecting comparisons of the type that was described in the previous paragraph, has been addressed in the statistics literature under the general term of "ranking & selection (r&s)" [31]. Some of the most general results for this problem work with the following basic positioning of it:

Assumption 1 The performance measure of interest for each entertained option $i = 1, \ldots, k$, is the unknown mean μ_i of a *normal* random variable (r.v.) X_i.

Assumption 2 Besides the means μ_i, the variances, σ_i, for the r.v.'s X_i, are also unknown and possibly unequal.

Assumption 3 It is possible to generate a sequence $\langle \hat{X}_{ij},\ j = 1, 2, \ldots \rangle$ of independent samples for each r.v. X_i

Assumption 4 The set of samples $\{\hat{X}_{ij},\ i = 1, \ldots, k\}$ – i.e., the samples obtained for the r.v.'s $X_i,\ i = 1, \ldots, k$, during the j-th round of sampling – can be independent or (positively) correlated.

Assumption 5 There is also a pre-specified parameter δ such that any pair of options $\{i, j\}$ with $|\mu_i - \mu_j| \leq \delta$ are treated as equivalent during the attempted comparison – in more technical terms, the parameter δ defines an *"indifference zone"* for the pursued comparison.

Assumption 6 Finally, all the existing approaches also allow for an erring probability a with $a < 1/k$.[3]

[3]The reader should notice that $1/k$ is the probability of selecting the correct option when this selection is performed completely randomly among the k available options.

Problem statement Under Assumptions 1-6, we want to design a sampling process and the accompanying inference logic that will select an option i in the indifference zone of $\arg\max_{j\in\{1...,k\}}\{\mu_j\}$ with probability $1-a$.

Of particular interest to this work, are solutions to the aforestated version of the r&s problem that are *"fully sequential"*. Generally speaking, these solutions go through a first sampling round that enable them to collect some information about the inherent variability in each r.v. X_i, and subsequently they perform an additional number of sampling rounds where the information that is obtained from the additional samples that are collected at each round is used to further assess the competitiveness of the options that are still entertained in that round, and potentially eliminate some of these options that are deemed to not be competitive anymore. The entire process terminates when the set of (remaining) competitive options becomes a singleton. Some perceived advantages of such a (fully) sequential procedure are: (i) the ability to make more expedient and efficient use of the information that is contained in the collected samples, eliminating early options that are not deemed competitive by this sampling process, and (ii) its amenability to implementation in a "real-time" operational setting.

Also, an additional potential feature of the r&s problem that was described in the previous paragraphs, is the presence of an option – to be denoted as the 0-th option – that is to be treated as the "preferred choice" as long as it belongs in the $\arg\max_{j\in\{0,1...,k\}}\{\mu_j\}$. The corresponding version of the r&s problem is known as "comparison with a standard" in the relevant literature [54, 30]. This version is particularly relevant in our case, since we want to alter the current policy only if the expected gain from the contemplated modification(s) is significant.

Finally, a more technical remark concerns the removal of the normality requirement for the r.v.'s $X_i,\ i=1,\ldots,k$, that was posed in Assumption 1. This can be attained by re-defining the j-th sample $\hat{X}_{ij}$ of the i-th random variable X_i as the average of N independent samples drawn from the corresponding distribution. Then, as long as the employed

sample size N is adequately large, the central limit theorem [66] ensures that this modified sample concept follows approximately a normal distribution with mean $E[X_i]$. Clearly, the practical significance of this remark is very important for the application of the existing r&s algorithms to various practical applications, including the application that is the focus of this work.

A fully sequential procedure that addresses the "comparison with a standard" version of the considered r&s problem under the aforestated Assumptions 1-6, is provided in [30]. Furthermore, from all the previous discussion, it is clear that this procedure defines a very effective tool for resolving the r&s problems that we need to address in this chapter. Therefore, we replicate this procedure in Figure 6.3, focusing on the particular variations that are of interest in this work, and we demonstrate its application to the particular context of the scheduling problem that is considered in this work, in the next section.

6.4 An empirical assessment of the proposed policy improving methods and some further implementational details

In the first parts of this section, we present a series of experiments that have the double purpose of (i) demonstrating and assessing the efficacy of the r&s algorithm of Figure 6.3 to detect improving opportunities for the FR-based scheduling policy of Chapters 4 and 5 for the considered CRLs, and also (ii) demonstrating and assessing the efficacy of the decisions that are effected by the FR-based scheduling policy of Chapters 4 and 5 at the most visited states of the underlying state space (and therefore, the most critical states for shaping the overall performance of the policy). All these experiments were conducted on a laptop with a 2.2 GHz i5 processor and 8GB of memory. The employed r&s algorithm was coded in C# and executed in Microsoft Visual Studio 2017.

The last part of the section builds upon the experience of the earlier parts in order to provide some ideas about the orchestration of the presented results in a full-fledged improving process for the FR-based scheduling policy (or any other deterministic scheduling

Setup: Select confidence level $1 - a$, indifference-zone parameter δ, and the first-stage sample size n_0. Also, set

$$\beta = \begin{cases} 1 - (1-a)^{1/k} & \text{if the obtained samples for the various options at each sampling iteration are independent} \\ a/k & \text{if the obtained samples for the various options at each sampling iteration are correlated} \end{cases}$$

and determine η by solving the equation

$$\big(1 + 2\eta\big)^{-(n_0-1)/2} = 2\beta$$

Initialization: Let $I = \{0, 1, 2, \ldots, k\}$ be the set of options in contention. Obtain n_0 observations $\hat{X}_{ij}, j = 1, 2, \ldots, n_0$, from each option $i = 0, 1, 2, \ldots, k$.
For all $i \neq l,\ l = 0, 1, 2, \ldots, k$, compute S_{il}^2, the sample variance of the difference between option i and option l, and let

$$a_{il} = \frac{\eta(n_0 - 1)S_{il}^2}{\delta_{il}} \quad \text{and} \quad \lambda_{il} = \frac{\delta_{il}}{2}$$

where

$$\delta_{il} = \begin{cases} \delta/2, & \text{if } i = 0 \ \vee l = 0 \\ \delta, & \text{o.w.} \end{cases}$$

Set the observation counter $r := n_0$ and go to *Screening*.
Screening: For each $i < l$, $i \in I$ and $l \in I$, if

$$\sum_{j=1}^{r} \big(\mathcal{X}_{ij} - \mathcal{X}_{lj}\big) \ \leq \ -\max\Big\{0,\ a_{il} - \lambda_{il} r\Big\},$$

then eliminate i from I; else if

$$\sum_{j=1}^{r} \big(\mathcal{X}_{ij} - \mathcal{X}_{lj}\big) \ \geq \ \max\Big\{0,\ a_{il} - \lambda_{il} r\Big\},$$

then eliminate l from I.
In the above inequalities,

$$\mathcal{X}_{ij} = \begin{cases} \hat{X}_{ij} + \delta/2, & \text{if } i = 0 \\ \hat{X}_{ij}, & \text{o.w.} \end{cases}$$

Stopping Rule: If $|I| = 1$, then stop and select the option whose index is in I. Otherwise, set $r = r + 1$, take one additional observation, $\hat{X}_{ir}$, from each option $i \in I$, and go to *Screening*.

Figure 6.3: The fully sequential procedure of [30] for resolving the "comparison with a standard" version of the r&s problem under Assumptions 1–6. It is assumed that the "standard" value μ_0 is unknown, and the parameter c has been set equal to 1.

policy that might provide a good starting solution to the considered scheduling problems).

6.4.1 Implementing the r&s algorithm of Figure 6.3 in an "on-line" operational mode

In this part of the presented experiments, we used the 20 CRL configurations that are listed in Table 4.2 in order to investigate empirically the ability of the r&s algorithm of Figure 6.3 to identify policy-improving opportunities, when this algorithm is implemented in an "on-line" operational mode for the underlying CRL. We remind the reader that, according to the corresponding discussion of Section 6.1, these improvements are sought locally at some pre-selected decision state $\mathbf{s}$, and in the case of an on-line operational mode, the primary guidance in this search is provided by the state potentials, $\hat{\mathbf{g}}(\mathbf{s}';\pi)$, of the states $\mathbf{s}'$ that are in the tangible reach $\mathcal{TR}(\mathbf{s})$ of state $\mathbf{s}$. In this experiment, the considered state $\mathbf{s}$ was the most visited state when the underlying CRL was operated in steady-state under the control of the corresponding FR-based scheduling policy.

Some particular details for this part of the experiment are as follows:

For each of the CRL configurations 1 to 16 in Table 4.2, we generated 30 problem instances by varying randomly the processing rates for the corresponding processing stages over the interval [1,10]. On the other hand, for each of the CRL configurations 17 to 20 of Table 4.2, we generated only 5 problem instances, with a similar range for the random processing rates of their processing stages, since the state spaces for these configurations are very large, and therefore, the computation of these state spaces and the performance evaluation of the corresponding scheduling policies took a very long time.

For each CRL instantiation that was generated in this experiment, we implemented the corresponding FR-based scheduling policy – to be denoted by π in the following – according to the logic described in Section 5.3. Then, we randomly selected one state $\mathbf{s}$ from the underlying state space, and we simulated 10,000 regenerative cycles of the line operation under policy π, in order to find the most visited decision state $\mathbf{s}_m$.

The data collected from these regenerative cycles were also used in order to obtain

an estimate, $\hat{g}(a(\widehat{\mathbf{s}_m;\pi});\pi)$, of the potential of the action $a(\mathbf{s}_m;\pi)$ that is selected at state $\mathbf{s}_m$ by policy π, according to the logic described in Section 6.2. We used the estimate $\hat{g}(a(\widehat{\mathbf{s}_m;\pi});\pi)$ in order to specify the indifference zone δ for the r&s algorithm; in particular, we considered two different values:

$$\delta = 0.001\hat{g}(a(\widehat{\mathbf{s}_m;\pi});\pi) \text{ and } \delta = 0.0005\hat{g}(a(\widehat{\mathbf{s}_m;\pi});\pi)$$

These values were expected to provide a very high discerning power for the applied r&s algorithm. In a similar spirit, we also set the erring probability a for this algorithm to the two values of

$$a = 0.05 \text{ and } a = 0.01$$

The combination of the above values for the parameters δ and a provides some information about the impact of these parameters on the computational effort and the execution time of the algorithm. At the same time, the employment of the really tight values for the parameters δ and a that were mentioned above, boosts the ability of the r&s algorithm to make a correct choice at every iteration of the experiment, and therefore, it allows us to obtain a good assessment of the ability of the original policy π to make a pertinent choice at the state $\mathbf{s}_m$ (i.e., the most visited state by this policy).

Finally, the initial 10,000 regenerative cycles for the CRL operation under policy π that were mentioned in the previous paragraphs, gave us also an estimate of the long-term throughput of the policy, $\widehat{\hat{\eta}(\pi)}$, according to the corresponding estimator that was introduced in Section 6.2. This estimate was used subsequently in the estimator of the state potentials under consideration, that is defined by Equation 6.24.

During the execution of the r&s algorithm of Figure 6.3 at the selected decision state $\mathbf{s}_m$, we used the average of a batch of 30 estimates obtained through Equation 6.24, as a single estimate for any of the compared potentials $\hat{g}(\mathbf{s};\pi),\ \mathbf{s} \in \mathcal{TR}(\mathbf{s}_m)$. This batching intended to address the normality requirement for the distributions of the various options

Table 6.1: The amount of sampling that is required by the "on-line" implementation of the r&s algorithm of Figure 6.3.

Config.	r	$\alpha = 5\%$		$\alpha = 1\%$	
		$\delta = 0.001\widehat{\tilde{\mathbf{g}}(\mathbf{s}_i;\pi)}$	$\delta = 0.0005\widehat{\tilde{\mathbf{g}}(\mathbf{s}_i;\pi)}$	$\delta = 0.001\widehat{\tilde{\mathbf{g}}(\mathbf{s}_i;\pi)}$	$\delta = 0.0005\widehat{\tilde{\mathbf{g}}(\mathbf{s}_i;\pi)}$
Config 1	Avg.	32976.4	72009.333	71086.1	201286.36
	Min.	2099	12186	7782	21350
	Max.	148275	280269	273920	821453
Config 2	Avg.	68024.5	156710.46	162662.16	480027.63
	Min.	8038	6758	20992	33280
	Max.	245350	483942	509235	1904486
Config 3	Avg.	47916.8	126287.5	106544.7	316015.73
	Min.	5427	2611	17357	29184
	Max.	146688	629453	410419	1137306
Config 4	Avg.	83224.966	196545.66	182637.83	596400.06
	Min.	8550	13005	40755	64307
	Max.	311296	599194	407398	2073242
Config 5	Avg.	95488.4	289512.3	220018.3	646735.36
	Min.	2304	63795	6298	107213
	Max.	401920	905933	861901	1949952
Config 6	Avg.	63631.93	133122.63	113645.6	393409.16
	Min.	6246	21453	5171	44032
	Max.	243610	612966	403917	1677978
Config 7	Avg.	83224.7	72009.2	220018.56	316015.36
	Min.	20224	17387	53116	76698
	Max.	477744	756985	843953	2536516
Config 8	Avg.	68024.43	196545.76	162662.83	480027.26
	Min.	16647	47691	39246	116374
	Max.	537969	1299023	735744	3033296
Config 9	Avg.	47916.366	196545.16	182637.73	393409.2
	Min.	11717	49723	45463	96990
	Max.	242521	992283	639194	2464795
Config 10	Avg.	95488.6	156710.3	162662.7	646735.6
	Min.	23239	40162	39246	157789
	Max.	434586	926385	732723	3197955
Config 11	Avg.	63631.1	72009.33	182637.9	646735.5
	Min.	15521	17387	44673	156384
	Max.	275538	773472	639194	3366711
Config 12	Avg.	63631.33	196545.03	113645.7	316015.96
	Min.	15489	49723	27527	78416
	Max.	529183	1006056	634689	2310007
Config 13	Avg.	83224.8	72009.566	220018.36	393409.8
	Min.	20066	18140	53499	96179
	Max.	314723	627961	1301937	2464795
Config 14	Avg.	83224.966	126287.8	182637.8	316015.73
	Min.	20191	30413	44673	78416
	Max.	313136	851767	772673	2536516
Config 15	Avg.	63631.73	133122.93	162662.66	393409.3
	Min.	15364	32807	39350	95585
	Max.	438559	546512	599245	2464795
Config 16	Avg.	83224.9	156710.76	113645.2	316015.03
	Min.	20316	38098	27527	76698
	Max.	411798	797361	1089191	2581982
Config 17	Avg.	204073.6	378860.4	660055.4	603858.8
	Min.	49254	92487	160932	157791
	Max.	1138975	2646077	2531860	7427441
Config 18	Avg.	286464	470129.6	319631.4	1180227
	Min.	69777	113144	77332	288538
	Max.	1506816	1781064	1870520	4824812
Config 19	Avg.	286464.6	378860.8	487987.2	1440082.6
	Min.	69501	91737	119636	349122
	Max.	1778688	2555300	2207232	9099889
Config 20	Avg.	254525.8	626838.2	426175.8	1920109.6
	Min.	61455	160648	103290	464242
	Max.	2116731	3771488	2889291	11640026

that are considered by the r&s algorithm of Figure 6.3.

Also, at each sampling iteration of the algorithm, we used the same regenerative cycles for the construction of the necessary estimates for all of the potentials that are still assessed during this iteration; hence, the potential estimates used during each sampling iteration were positively correlated.

Table 6.1 reports the average, minimum and maximum number of sampling cycles that

were required by the algorithm in order to reach a decision, for all the considered CRL configurations. We see that these numbers can be quite significant. On the other hand, the execution time of the algorithm on any particular instantiation of the considered CRLs lasted from a few seconds for the smaller configurations, to no more than 30 minutes for the largest ones. Table 6.1 also reveals that as the employed values for the parameters δ and a get tighter, the required amount of sampling increases, but, for all the selected values of these parameters, the number of the required sampling cycles has the same order of magnitude.

Finally, we should also notice that when we relaxed the indifference zone δ by an order of magnitude, to a value of $0.01\hat{g}(a(\widehat{\mathbf{s}_m;\pi});\pi)$, the corresponding values to those reported in Table 6.1 dropped drastically, never exceeding the value of 1000 even for the largest configurations.

In order to assess the quality of the decisions that were made by the r&s algorithm of Figure 6.3 upon its execution on the considered CRL instantiations, we also evaluated the relative gains in throughput that are incurred by the policies π' that are specified by the actions that are selected by this algorithm. More specifically, for each CRL instantiation in the considered experiment, we computed the quantity

$$\%\text{-}\mathit{improvement} = \frac{\hat{\eta}(\pi') - \hat{\eta}(\pi)}{\hat{\eta}(\pi)} \times 100$$

where the quantities $\hat{\eta}(\pi')$ and $\hat{\eta}(\pi)$ that appear in this expression, were computed through the solution of the corresponding Poisson equation [57]. Table 6.2 tabulates the obtained results, reporting the average, minimum and maximum %-improvements that were observed for the generated instances from the 20 CRL configurations of Table 4.2.

The perusal of the values that are reported in Table 6.2 reveals that, under the employed parameterization, the considered r&s algorithm never came up with a decision that would result in a degraded performance with respect to the applied policy π. Furthermore, the

values that are reported in Table 6.2 are very small, a fact that indicates that the FR-based scheduling policy typically makes pretty good choices in the most visited state $\mathbf{s}_m$. In fact, it is also true that in most of the considered CRL instantiations, the action selected with the r&s algorithm coincided with the action that was selected by the FR-based scheduling policy itself. This finding is consistent with the experimental results of Chapters 4 and 5 that have indicated that the FR-based scheduling policy will make a pertinent decision at most decision states.

6.4.2 Implementing the r&s algorithm of Figure 6.3 in an "off-line" operational mode

In this part of our experiments, we shifted attention to an "off-line" determination of a best action at the decision state under consideration. The basic set up of this experiment was similar to that used in the "on-line" case; in particular, we still used the same number of instantiations from the same set of CRL configurations that are reported in Table 4.2, and, in each case, we focused on the same state $\mathbf{s}_m$, that is the most frequently visited state by the corresponding FR-based scheduling policy. Also, the selection of the employed values for the parameters δ and a of the r&s algorithm was made in the same manner as in the "on-line" case, but for parameter δ we used the estimate $\widehat{\hat{\eta}(\pi)}$ as reference. And we used a batching scheme of 30 samples per batch in order to obtained the required samples by the r&s algorithm.

As discussed in the closing part of Section 6.1, in the "off-line" case, the selection of the best action at the considered decision point $\mathbf{s}_m$ should be based on a direct comparison of the throughputs $\hat{\eta}(\pi')$ of the corresponding policies π' that are defined by each state $\mathbf{s}' \in \mathcal{TR}(\mathbf{s}_m)$. The required estimates $\widehat{\hat{\eta}(\pi')}$, for each policy π', were obtained through the corresponding estimator of Section 6.2, while simulating the system operation under this policy. As a result, in this case, the throughput estimates computed for each competing option at each sampling cycle of the algorithm were independent from each other, and the r&s algorithm of Figure 6.3 was configured accordingly.

Table 6.2: An empirical assessment of the relative throughput gain that is incurred by the policy π' that is recommended by the r&s algorithm of Figure 6.3, under an "on-line" execution of this algorithm.

Config.	%improvement	$\alpha = 5\%$		$\alpha = 1\%$	
		$\delta = 0.001\widehat{\eta(\pi)}$	$\delta = 0.0005\widehat{\eta(\pi)}$	$\delta = 0.001\widehat{\eta(\pi)}$	$\delta = 0.0005\widehat{\eta(\pi)}$
Conf 1	Avg.	0	0	0	0
	Min.	0	0	0	0
	Max.	0	0	0	0
Conf 2	Avg.	0	0	0	0
	Min.	0	0	0	0
	Max.	0	0	0	0
Conf 3	Avg.	0.0077	0.0077	0.0077	0.0077
	Min.	0	0	0	0
	Max.	0.1198	0.1198	0.1198	0.1198
Conf 4	Avg.	0.0116	0.0116	0.0116	0.0116
	Min.	0	0	0	0
	Max.	0.1637	0.1637	0.1637	0.1637
Conf 5	Avg.	0.0070	0.0070	0.0070	0.0070
	Min.	0	0	0	0
	Max.	0.2121	0.2121	0.2121	0.2121
Conf 6	Avg.	0.0155	0.0155	0.0155	0.0155
	Min.	0	0	0	0
	Max.	0.2321	0.2321	0.2321	0.2321
Conf 7	Avg.	0.0296	0.0296	0.0296	0.0296
	Min.	0	0	0	0
	Max.	0.3426	0.3426	0.3426	0.3426
Conf 8	Avg.	0.0042	0.0042	0.0042	0.0042
	Min.	0	0	0	0
	Max.	0.0755	0.0755	0.0755	0.0755
Conf 9	Avg.	0.0011	0.0011	0.0011	0.0011
	Min.	0	0	0	0
	Max.	0.0339	0.0339	0.0339	0.0339
Conf 10	Avg.	0.0257	0.0257	0.0257	0.0257
	Min.	0	0	0	0
	Max.	0.1661	0.1661	0.1661	0.1661
Conf 11	Avg.	0.0079	0.0079	0.0079	0.0079
	Min.	0	0	0	0
	Max.	0.1407	0.1407	0.1407	0.1407
Conf 12	Avg.	0.0093	0.0093	0.0093	0.0093
	Min.	0	0	0	0
	Max.	0.0851	0.0851	0.0851	0.0851
Conf 13	Avg.	0.0114	0.0114	0.0114	0.0114
	Min.	0	0	0	0
	Max.	0.1846	0.1846	0.1846	0.1846
Conf 14	Avg.	0.0101	0.0101	0.0101	0.0101
	Min.	0	0	0	0
	Max.	0.0796	0.0796	0.0796	0.0796
Conf 15	Avg.	0.0098	0.0098	0.0098	0.0098
	Min.	0	0	0	0
	Max.	0.1171	0.1171	0.1171	0.1171
Conf 16	Avg.	0.0120	0.0120	0.0120	0.0120
	Min.	0	0	0	0
	Max.	0.1058	0.1058	0.1058	0.1058
Conf 17	Avg.	0.0258	0.0258	0.0258	0.0258
	Min.	0	0	0	0
	Max.	0.1294	0.1294	0.1294	0.1294
Conf 18	Avg.	0	0	0	0
	Min.	0	0	0	0
	Max.	0	0	0	0
Conf 19	Avg.	0.0155	0.0155	0.0155	0.0155
	Min.	0	0	0	0
	Max.	0.0779	0.0779	0.0779	0.0779
Conf 20	Avg.	0	0	0	0
	Min.	0	0	0	0
	Max.	0	0	0	0

The experimental results obtained for the "off-line" case are reported in Tables 6.3 and 6.4, that are the counterparts of Tables 6.1 and 6.2 of the previous subsection. Some observations that can be made about these results are as follows:

When it comes to the amount of the required sampling, the juxtaposition of Tables 6.1 and 6.3 indicates that, in the "off-line" case, the amount of the sampling that is employed

Table 6.3: The amount of sampling that is required by the "off-line" implementation of the r&s algorithm of Figure 6.3.

Config.	r	$\alpha = 5\%$		$\alpha = 1\%$	
		$\delta = 0.001\widehat{\eta(\pi)}$	$\delta = 0.0005\widehat{\eta(\pi)}$	$\delta = 0.001\widehat{\eta(\pi)}$	$\delta = 0.0005\widehat{\eta(\pi)}$
Config 1	Avg.	20610.366	45006.53	44429.4	125804.03
	Min.	1312	7616	4864	13344
	Max.	92672	175168	171200	513408
Config 2	Avg.	42515.366	97943.1	101664.46	300017.66
	Min.	5024	4224	13120	20800
	Max.	153344	302464	318272	1190304
Config 3	Avg.	29948.566	78929.166	66590.83	197509.1
	Min.	3392	1632	10848	18240
	Max.	91680	393408	256512	710816
Config 4	Avg.	52015.666	122841.96	114148.26	372750.6
	Min.	5344	8128	25472	40192
	Max.	194560	374496	254624	1295776
Config 5	Avg.	59680.2	180945.66	137511.03	404209.8
	Min.	1440	39872	3936	67008
	Max.	251200	566208	538688	1218720
Config 6	Avg.	39770.53	83201.5	71028.2	245881.5
	Min.	3904	13408	3232	27520
	Max.	152256	383104	252448	1048736
Config 7	Avg.	52015.43	45006.13	137511.2	197509.36
	Min.	11184	9328	28475	42171
	Max.	246575	428110	389959	1387813
Config 8	Avg.	42515.13	122841.1	101664.5	300017.1
	Min.	9508	26194	20979	63651
	Max.	293715	689049	358176	1595793
Config 9	Avg.	29948.9	122841.8	114148.06	245881.1
	Min.	6668	32543	27924	57215
	Max.	121628	497337	285348	1294617
Config 10	Avg.	59680.266	97943.73	101664.6	404209.26
	Min.	12941	27563	20979	88880
	Max.	211936	481047	356288	1594513
Config 11	Avg.	39770.466	45006.766	114148.9	404209.06
	Min.	8735	9328	25454	84490
	Max.	132442	438414	285348	1699985
Config 12	Avg.	39770.9	122841.83	71028.83	197509.13
	Min.	8632	32543	14993	47540
	Max.	290970	505945	325652	1246245
Config 13	Avg.	52015.93	45006	137511.3	245881.13
	Min.	10691	11683	29672	54680
	Max.	144687	347470	676199	1294617
Config 14	Avg.	52015.33	78929.466	114148.06	197509.83
	Min.	11081	16112	25454	47540
	Max.	143695	453425	368772	1387813
Config 15	Avg.	39770.3	83201.866	101664.56	245881.2
	Min.	8242	19322	21306	52824
	Max.	234330	258369	272864	1294617
Config 16	Avg.	52015.9	97943.1	71028.5	197509.43
	Min.	11472	21112	14993	42171
	Max.	205359	400407	609716	1416229
Config 17	Avg.	127546.6	236787.8	412534.2	377411.8
	Min.	26373	52234	90379	115687
	Max.	584314	1417011	1169878	4264739
Config 18	Avg.	179040.8	293830.2	199770.4	737642.4
	Min.	39014	59745	41893	164040
	Max.	762720	819334	969306	2277866
Config 19	Avg.	179040	236787.8	304992.4	900051.8
	Min.	38150	49892	68870	190954
	Max.	932640	1360275	1074528	4787379
Config 20	Avg.	159078.6	391774.2	266359.6	1200068.6
	Min.	32968	110252	56421	250689
	Max.	1163878	1965406	1539447	6074948

by the r&s algorithm during its execution on each CRL instantiation, tends to be less than the corresponding amount of sampling that is observed in the "on-line" case; but all these amounts are of the same order of magnitude.

On the other hand, the juxtaposition of Tables 6.2 and 6.4 reveals that these two tables are identical; i.e., for each considered CRL instantiation, the employed r&s algorithm has selected the same action at the corresponding state $\mathbf{s}_m$ in, both, the "on-line" and the "off-

Table 6.4: An empirical assessment of the relative throughput gain that is incurred by the policy π' that is recommended by the r&s algorithm of Figure 6.3, under an "off-line" execution of this algorithm.

Config.	%improvement	$\alpha = 5\%$		$\alpha = 1\%$	
		$\delta = 0.001\widehat{\eta(\pi)}$	$\delta = 0.0005\widehat{\eta(\pi)}$	$\delta = 0.001\widehat{\eta(\pi)}$	$\delta = 0.0005\widehat{\eta(\pi)}$
Conf 1	Avg.	0	0	0	0
	Min.	0	0	0	0
	Max.	0	0	0	0
Conf 2	Avg.	0	0	0	0
	Min.	0	0	0	0
	Max.	0	0	0	0
Conf 3	Avg.	0.0077	0.0077	0.0077	0.0077
	Min.	0	0	0	0
	Max.	0.1198	0.1198	0.1198	0.1198
Conf 4	Avg.	0.0116	0.0116	0.0116	0.0116
	Min.	0	0	0	0
	Max.	0.1637	0.1637	0.1637	0.1637
Conf 5	Avg.	0.0070	0.0070	0.0070	0.0070
	Min.	0	0	0	0
	Max.	0.2121	0.2121	0.2121	0.2121
Conf 6	Avg.	0.0155	0.0155	0.0155	0.0155
	Min.	0	0	0	0
	Max.	0.2321	0.2321	0.2321	0.2321
Conf 7	Avg.	0.0296	0.0296	0.0296	0.0296
	Min.	0	0	0	0
	Max.	0.3426	0.3426	0.3426	0.3426
Conf 8	Avg.	0.0042	0.0042	0.0042	0.0042
	Min.	0	0	0	0
	Max.	0.0755	0.0755	0.0755	0.0755
Conf 9	Avg.	0.0011	0.0011	0.0011	0.0011
	Min.	0	0	0	0
	Max.	0.0339	0.0339	0.0339	0.0339
Conf 10	Avg.	0.0257	0.0257	0.0257	0.0257
	Min.	0	0	0	0
	Max.	0.1661	0.1661	0.1661	0.1661
Conf 11	Avg.	0.0079	0.0079	0.0079	0.0079
	Min.	0	0	0	0
	Max.	0.1407	0.1407	0.1407	0.1407
Conf 12	Avg.	0.0093	0.0093	0.0093	0.0093
	Min.	0	0	0	0
	Max.	0.0851	0.0851	0.0851	0.0851
Conf 13	Avg.	0.0114	0.0114	0.0114	0.0114
	Min.	0	0	0	0
	Max.	0.1846	0.1846	0.1846	0.1846
Conf 14	Avg.	0.0101	0.0101	0.0101	0.0101
	Min.	0	0	0	0
	Max.	0.0796	0.0796	0.0796	0.0796
Conf 15	Avg.	0.0098	0.0098	0.0098	0.0098
	Min.	0	0	0	0
	Max.	0.1171	0.1171	0.1171	0.1171
Conf 16	Avg.	0.0120	0.0120	0.0120	0.0120
	Min.	0	0	0	0
	Max.	0.1058	0.1058	0.1058	0.1058
Conf 17	Avg.	0.0258	0.0258	0.0258	0.0258
	Min.	0	0	0	0
	Max.	0.1294	0.1294	0.1294	0.1294
Conf 18	Avg.	0	0	0	0
	Min.	0	0	0	0
	Max.	0	0	0	0
Conf 19	Avg.	0.0155	0.0155	0.0155	0.0155
	Min.	0	0	0	0
	Max.	0.0779	0.0779	0.0779	0.0779
Conf 20	Avg.	0	0	0	0
	Min.	0	0	0	0
	Max.	0	0	0	0

line" setups. We consider this fact as a manifestation of the high discerning power and the robustness of the employed r&s algorithm, especially under the pricing of the parameters δ and a that was described in the previous parts of this section.

Closing the discussion on the experiments that are presented in this section, we also

Table 6.5: The relative absolute error in the throughput estimates that were used in the potential estimator of Equation 6.24.

Config.		$\frac{\lvert\hat{\eta}(\pi)-\widehat{\hat{\eta}(\pi)}\rvert}{\hat{\eta}(\pi)}$	Config.		$\frac{\lvert\hat{\eta}(\pi)-\widehat{\hat{\eta}(\pi)}\rvert}{\hat{\eta}(\pi)}$
Config 1	Avg.	0.00052	Config 11	Avg.	0.02715
	Min.	1.10E-05		Min.	0.00044
	Max.	0.0017		Max.	0.0769
Config 2	Avg.	0.0067	Config 12	Avg.	0.0171
	Min.	4.86E-05		Min.	0.00047
	Max.	0.0339		Max.	0.0488
Config 3	Avg.	0.0249	Config 13	Avg.	0.023
	Min.	0.00011		Min.	0.00042
	Max.	0.0636		Max.	0.0805
Config 4	Avg.	0.0189	Config 14	Avg.	0.0146
	Min.	8.15E-06		Min.	0.00086
	Max.	0.0790		Max.	0.0300
Config 5	Avg.	0.0094	Config 15	Avg.	0.03053
	Min.	8.76E-05		Min.	0.011
	Max.	0.0690		Max.	0.07584
Config 6	Avg.	0.0044	Config 16	Avg.	0.02828
	Min.	0.00012		Min.	0.00338
	Max.	0.0178		Max.	0.07249
Config 7	Avg.	0.0401	Config 17	Avg.	0.0414
	Min.	0.0004		Min.	0.001
	Max.	0.110		Max.	0.10582
Config 8	Avg.	0.0455	Config 18	Avg.	0.05074
	Min.	0.00028		Min.	0.01359
	Max.	0.1412		Max.	0.09187
Config 9	Avg.	0.0039	Config 19	Avg.	0.011756
	Min.	4.84E-05		Min.	0.00257
	Max.	0.0233		Max.	0.0210
Config 10	Avg.	0.05712	Config 20	Avg.	0.0095
	Min.	0.0065		Min.	0.0014
	Max.	0.1534		Max.	0.01925

report, in Table 6.5, the relative absolute error

$$\frac{|\hat{\eta}(\pi) - \widehat{\hat{\eta}(\pi)}|}{\hat{\eta}(\pi)}$$

in the throughput estimates that we have used in the potential estimator of Equation 6.24, during the "on-line" implementation of the r&s algorithm of Figure 6.3. The perusal of this table reveals that, at least for certain configurations, these errors can be considerable, in a few cases even reaching or exceeding a value of 10%. Nevertheless, the findings that were reported in the earlier parts of this section about the observed performance of the considered algorithm, suggest that these errors in the employed throughput estimates do not have a significant negative impact on the algorithm performance.

6.4.3 Orchestrating the presented developments into a policy-improving mechanism

In the last part of this chapter we provide a few remarks on how to orchestrate the developments that were presented in its earlier parts, into a complete mechanism able to support a systematic improvement of the FR-based scheduling policy for the RAS classes that have

been considered in this work. Furthermore, it should be clear that this discussion, and also all the earlier developments in this chapter, apply not only to the aforementioned policy, but also to any other heuristic deterministic policy for the considered environments that is expected to have a good bottomline performance.

The proposed policy-improving mechanism is essentially an iterative search process. Starting with the original heuristic policy, at each iteration the proposed search scheme will select one or more decision states and it will assess the efficacy of the decisions that are effected at those states by the current policy π. More specifically, the selection of a set $\tilde{X} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n\}$ of decision states to be assessed at a given iteration, defines a set of potential modifications to the current policy π which is succinctly described by the set $\mathcal{TR}(\mathbf{s}_1) \times \mathcal{TR}(\mathbf{s}_2) \times \ldots \times \mathcal{TR}(\mathbf{s}_n)$; each element of this set is a n-tuple of tangible states that defines a modified policy π' through the replacement of the decision of policy π at each state $\mathbf{s}_i \in \tilde{X}$ with the corresponding decision (i.e., tangible state) that is contained in this tuple. Collectively, all policies π' that are defined in this way define a "local neighborhood" of policy π that must be searched for the best policy. The mechanism for this local search is the r&s algorithm that was presented in the previous parts of this chapter.

Clearly, the cardinality of the set $\tilde{X}$ to be used at each iteration will affect the size of the induced local neighborhood, and thus the complexity of the corresponding search process. Also, the set of decision states to be included in the aforementioned sets $\tilde{X}$ will affect the performance of this search scheme. In general, most visited states by the current policy π constitute good candidates for the reasons that were explained in the earlier parts of this section. But it is also true that if the starting heuristic policy is of high quality, then, these states might be less likely to generate actual improvements for the current policy, since the corresponding decisions might be already optimized. On the other hand, improving the decision at a decision state that is visited with a low frequency might still not have a major impact on the overall performance of the underlying system, to the point that it might not be worth considering. Clearly, a more thorough understanding of the pertinent composition of

the set $\tilde{X}$ is necessary, and this is a problem that would define an interesting line for future research.

On the representational side, the proposed search mechanism will need to explicitly store the identified decisions that will differ from the decisions that are effected by the original heuristic policy. Based on all the previous discussion, it is expected that, if the original policy is of good quality (as is the case with the FR-based scheduling policy that has been developed in this thesis), then, the decision points where this policy will need to be corrected might not be that many; this is especially true when we also take into consideration the above discussion about the potential insignificance of less visited states in the overall performance of the derived policy. But, more generally, the form and the amount of storage space that should be provided for tracing the effected modifications to the original policy should be considered as an additional "parameter" of the presented scheme that should be determined before its implementation.

A last theme that can be further discussed in the context of the proposed search mechanism, is the selection of the parameters δ and a for the employed r&s algorithm. As we saw in the previous section, selecting some tight values for these two parameters will establish a high discerning power for the algorithm, and a high quality for its decisions, but it will also result in a requirement for a very large amount of samples from each evaluated option. In view of this realization, it might be pertinent to start the proposed search by using some more relaxed values for these parameters, and tighten them as the overall search progresses to more difficult policy neighborhoods. This remark is especially pertinent for the parameter δ, that defines the indifference zone utilized by the algorithm, for the following two reasons: (i) The observation in our reported experiments that a relaxation of the employed value for δ can lead to a very drastic reduction of the required sampling. (ii) The fact that the r&s algorithm of Figure 6.3 will favor the incumbent action when it has to choose within the indifference zone, which guards against an unintentional degradation of the maintained performance with respect to the performance of the original policy. Hence,

by starting with a larg(er) value for δ, the proposed search scheme will be able to identify fast any possible improvements that will have a drastic impact on the performance of the underlying system, and once this lower value cannot effect any further improvement, one can consider decreasing it and conducting a new search starting with the policy π' that was obtained in the first phase. A pertinent calibration and evolution of these values is another issue that is open to future investigation.

Closing our discussion on the considered policy-improving scheme, we want also to make the following two remarks:

First, the above description of the proposed policy-improving scheme also reveals the affinity of this scheme to the policy-iteration (PI) algorithm of the classical MDP theory [57]. But in the proposed scheme, policies are evaluated and improved in a more incremental and localized manner, in an effort to control the underlying representational and computational complexities. Thus, from a more theoretical standpoint, the presented results expose the "flexibility" that is inherent in the mechanisms that enable the PI algorithm in the first place.

Furthermore, the presented policy-improving scheme defines also an interesting and novel proposition at the representational level, in the way that it leverages the original heuristic policy (more specifically, in the context of this thesis, the FR-based scheduling policy) as a means for encoding the decisions to be effected at most decision states of the underlying MDP. At the same time, improved decisions that are incurred by the search mechanism described in the previous parts of this section, can be stored in a more tabular manner. We believe that this "distribution" of the overall policy logic over a number of representational modes that complement each other, in an effort to ensure, both, the operational efficiency and the compactness of the resulting policy, is an unexplored territory that might hold more extensive potential for hard decision-making problems like those addressed in this work, and, therefore, it might deserve more extensive attention.

CHAPTER 7
SUMMARY OF THE MAJOR CONTRIBUTIONS AND POSSIBLE FUTURE EXTENSIONS

7.1 The major contributions

In the previous chapters of this work we have provided a complete framework for the real-time management of complex sequential resource allocation systems with blocking and deadlocking effects in their dynamics. In particular, this framework is a detailed realization of the integrated event-driven control framework depicted in Figure 1.1.

Hence, in Chapter 3, we leveraged some formal DES-based representations of the RAS behavior and we introduced a new DAP class for the considered sequential RAS that was characterized as the class of "maximal linear" DAPs. We also provided a complete algorithm for enumerating all the elements of this class for any given D/C-RAS instance Φ. Finally, we have presented some numerical experimentation that demonstrated the efficacy of the presented algorithm.

In Chapter 4, we have provided a scheduling methodology that aims to maximize the throughput of complex RAS with blocking and deadlocking effects. This methodology is based on the solution of a pertinent "fluid" relaxation of the addressed scheduling problem, and it is enabled by a pre-established ability to control the underlying RAS for deadlock freedom, and by the further ability to express the corresponding DAP as a set of linear inequalities on the system state. However, in this chapter, the detailed specification and parameterization of the developed fluid model was performed in an *ad hoc* manner.

Acknowledging this particular element in the developments of Chapter 4, in Chapter 5 we tried to strengthen and (further) formalize these developments by taking advantage of the representational and analytical capabilities of the Petri net modeling framework, which

is one of the main formal representational frameworks employed by the current DES theory. These capabilities have enabled a seamless treatment of the behavioral and the time-based dynamics of the underlying RAS, and they also support a notion of "fluidization" of these dynamics through the more recent developments in the area of timed and untimed continuous PN models; this last capability was especially critical for the systematic derivation of the sought "fluid relaxation" models and formulations. From a more conceptual standpoint, the "fluid" models that are presented in Chapters 4 and 5, when combined with the "linear" deadlock avoidance policies that have been employed in this work, provide a complete and very efficient implementation of the DES-based controller for complex RAS that is depicted in Figure 1.1.

Finally, in Chapter 6, we presented an extension to the developments of Chapters 4 and 5, in the form of a "correction" algorithm that aims to detect suboptimal decisions by the FR-based scheduling policy and correct them. These "corrections" can be effected either in an "off-line" mode, by simulating the dynamics of the underlying RAS, or in an "on-line" mode where the underlying RAS is fully operational and the necessary corrections are inferred from the observed behavior of the system. In both of these modes, and especially the second one, the developments of this chapter endow the control framework depicted in Figure 1.1 with a "learning" capability. From a more methodological standpoint, the results that were developed in this chapter are based on the sensitivity analysis of Markov reward processes and the r&s algorithm developed by [31]. The chapter also contains a series of numerical results that demonstrate and assess the efficacy of the developed methodology.

7.2 Possible future extensions

In this section, we briefly discuss some potential extensions of this work.

Along these lines, perhaps the currently most open and interesting direction for further exploration is the systematic development of the policy-improving mechanism that was

outlined in Section 6.4.3. The discussion on this topic that was provided in Section 6.4.3, defines a good starting "base" for any further exploration in this direction.

A second issue that is quite intellectually stimulating, and it can also have significant practical implications for an even more effective implementation of the presented developments, is the more thorough and more complete understanding of the mechanisms that incur the sub-optimality of the FR-based scheduling policy. We provided some pointers in this direction in the closing part of Chapter 5, but more extensive and systematic work can be done in this area.

Finally, a third possibility for extending the results of this work, is through the adaptation of the notion of "robustness" that has been pursued in the context of the FR-based scheduling in [7], to the particular scheduling problem and the methods that have been pursued in this work. Generally speaking, the approach of [7] seeks to bring the effects of the randomness that are experienced in the processing times and / or the arrival processes of the underlying stochastic network, into the formulated relaxing LP, by leveraging techniques borrowed from the area of robust linear programming. In [7], it is reported that this approach leads to improved scheduling policies compared to the scheduling policies that are obtained through the more basic implementation of the FR-based scheduling method, using more standard LP formulations. It would be interesting to investigate the implementational details and the potential effects of this approach in the context of the RAS throughput maximization problems that have been considered in this work.

Appendices

APPENDIX A

A BRIEF INTRODUCTION TO PETRI NET MODELING THEORY

This appendix provides an overview of the Petri net (PN) modeling framework, and introduces the relevant notation that is used in the rest of this document. In the subsequent discussion, we focus on those concepts and results of the general PN theory that are necessary for the formal modeling of the workflows of the CRL model that is considered in this document, and for the definition and the analysis of some basic properties of these workflows; a more expansive exposition of the basic PN theory can be found in [49].

A formal definition of the basic Petri net model is as follows:

Definition 15 *[49] A* Petri net (PN) system *is defined by a quadruple* $\mathcal{N} = \langle P, T, W, \mathbf{m}_0 \rangle$*, where*

- P *is the set of* places*,*
- T *is the set of* transitions*,*
- $W : (P \times T) \cup (T \times P) \to \mathbb{Z}_0^+$ *is the* flow relation*, and*
- $\mathbf{m}_0 : P \to \mathbb{Z}_0^+$ *is the net* initial marking*, assigning to each place* $p \in P$*,* $\mathbf{m}_0[p]$ tokens*.*

The first three items in Definition 15 essentially define a *weighted bipartite digraph* representing the system *structure* that generates the corresponding DES dynamics. The last item defines the system *initial state*. A conventional graphical representation of the net structure and its marking depicts nodes corresponding to places by empty circles, nodes corresponding to transitions by bars, and the tokens located at the various places by small filled circles. The flow relation W is depicted by directed edges that link every nodal pair

for which the corresponding W-value is non-zero. These edges point from the first node of the corresponding pair to the second, and they are also labelled – or, *"weighted"* – by the corresponding W-value. By convention, absence of a label for any edge implies that the corresponding W-value is equal to unity.

PN structure-related concepts and properties: Given a transition $t \in T$, the set of places p for which $(p,t) > 0$ (resp., $(t,p) > 0$) is known as the set of *input* (resp., *output*) places of t. Similarly, given a place $p \in P$, the set of transitions t for which $(t,p) > 0$ (resp., $(p,t) > 0$) is known as the set of *input* (resp., *output*) transitions of p. It is customary in the PN literature to denote the set of input (resp., output) transitions of a place p by ${}^\bullet p$ (resp., $p^\bullet$). Similarly, the set of input (resp., output) places of a transition t is denoted by ${}^\bullet t$ (resp., $t^\bullet$). This notation is also generalized to any set of places or transitions, X, e.g. ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$.

The sequence $X = \langle x_1 \ldots x_n \rangle \in (P \cup T)^*$ is a *path*, if and only if (*iff*) $x_{i+1} \in x_i^\bullet, i = 1, \ldots, n-1$. A path X is *simple iff* there is no repetition of nodes in it. Furthermore, a path X is characterized as a *circuit iff* $x_1 \equiv x_n$.

A PN with a flow relation W mapping onto $\{0,1\}$ is said to be *ordinary*. If only the restriction of W to $(P \times T)$ maps on $\{0,1\}$, the PN is said to be PT*-ordinary*. An ordinary PN s.t. $\forall t \in T$, $|t^\bullet| = |{}^\bullet t| = 1$, is characterized as a *state machine*, while an ordinary PN s.t. $\forall p \in P$, $|p^\bullet| = |{}^\bullet p| = 1$, is characterized as a *marked graph*.

A PN is said to be *pure* if $\forall (x,y) \in (P \times T) \cup (T \times P), W(x,y) > 0 \Rightarrow W(y,x) = 0$. The flow relation of pure PNs can be represented by the *flow matrix* $\Theta = \Theta^+ - \Theta^-$ where $\Theta^+[p,t] = W(t,p)$ and $\Theta^-[p,t] = W(p,t)$.

PN dynamics-related concepts and properties: In the PN modelling framework, the system state is represented by the net *marking* $\mathbf{m}$; this is a nonnegative integer vector of dimensionality $|P|$, that is perceived as a function from P to $\mathbb{Z}_0^+$, assigning a *token* content to each place $p \in P$. The net marking $\mathbf{m}$ is initialized to marking $\mathbf{m}_0$, introduced

in Definition 15, and it subsequently evolves through a set of rules summarized in the concept of *transition firing*. A concise characterization of this concept is as follows: Given a marking $\mathbf{m}$, a transition t is *enabled iff* for every place $p \in {}^\bullet t$, $\mathbf{m}[p] \geq W(p,t)$, and this is denoted by $\mathbf{m}[t\rangle$. On the other hand, a transition $t \in T$ is *disabled* by a place $p \in {}^\bullet t$ at $\mathbf{m}$ *iff* $\mathbf{m}[p] < W(p,t)$. Furthermore, a place $p \in P$ for which there exists $t \in p^\bullet$ s.t. $\mathbf{m}[p] < W(p,t)$, is a *disabling* place at $\mathbf{m}$. Given a marking $\mathbf{m}$, a transition t can be *fired* only if it is enabled at $\mathbf{m}$; the firing of such an enabled transition t results in a new marking $\mathbf{m}'$, which is obtained from $\mathbf{m}$ by removing $W(p,t)$ tokens from each place $p \in {}^\bullet t$, and placing $W(t,p')$ tokens in each place $p' \in t^\bullet$. For pure PNs, the marking evolution incurred by the firing of a transition t can be concisely expressed by the *state equation*:

$$\mathbf{m}' = \mathbf{m} + \Theta \cdot \mathbf{1}_t \tag{A.1}$$

where $\mathbf{1}_t$ denotes the unit vector of dimensionality $|T|$ and with the unit element located at the component corresponding to transition t.

The set of markings reachable from the initial marking $\mathbf{m}_0$ through any *fireable* sequence of transitions is denoted by $R(\mathcal{N}, \mathbf{m}_0)$ and it is referred to as the net *reachability space*. In the case of pure PNs, Eq. A.1 implies that a necessary reachability condition for any given marking $\mathbf{m}$ (i.e., for $\mathbf{m} \in R(\mathcal{N}, \mathbf{m}_0)$), is that the following system of equations is feasible in $\mathbf{z}$:

$$\mathbf{m} = \mathbf{m}_0 + \Theta \cdot \mathbf{z} \tag{A.2}$$

$$\mathbf{z} \in (\mathbb{Z}_0^+)^{|T|} \tag{A.3}$$

A PN $\mathcal{N} = (P, T, W, \mathbf{m}_0)$ is said to be *bounded iff* all markings $\mathbf{m} \in R(\mathcal{N}, \mathbf{m}_0)$ are bounded; i.e., for every place $p \in P$, $\exists B(p) \in \mathbb{Z}_0^+$ s.t. $\mathbf{m}[p] \leq B(p)$, $\forall \mathbf{m} \in R(\mathcal{N}, \mathbf{m}_0)$. $\mathcal{N}$ is said to be *structurally bounded iff* it is bounded for any initial marking $\mathbf{m}_0$. $\mathcal{N}$ is said to be *reversible iff* $\mathbf{m}_0 \in R(\mathcal{N}, \mathbf{m})$, for all $\mathbf{m} \in R(\mathcal{N}, \mathbf{m}_0)$. A transition $t \in T$ is said to be

live iff for all $\mathbf{m} \in R(\mathcal{N}, \mathbf{m}_0)$, there exists $\mathbf{m}' \in R(\mathcal{N}, \mathbf{m})$ s.t. $\mathbf{m}'[t\rangle$; non-live transitions are said to be *dead* at those markings $\mathbf{m} \in R(\mathcal{N}, \mathbf{m}_0)$ for which there is no $\mathbf{m}' \in R(\mathcal{N}, \mathbf{m})$ s.t. $\mathbf{m}'[t\rangle$. PN $\mathcal{N}$ is *quasi-live iff* for all $t \in T$, there exists $\mathbf{m} \in R(\mathcal{N}, \mathbf{m}_0)$ s.t. $\mathbf{m}[t\rangle$; it is *weakly live iff* for all $\mathbf{m} \in R(\mathcal{N}, \mathbf{m}_0)$, there exists $t \in T$ s.t. $\mathbf{m}[t\rangle$; and it is *live iff* for all $t \in T$, t is live. A marking $\mathbf{m} \in R(\mathcal{N}, \mathbf{m}_0)$ is a (total) *deadlock iff* every $t \in T$ is dead at $\mathbf{m}$.[1]

PN invariants and semiflows: As suggested by their name, PN invariants provide an analytical characterization of various notions of *invariance* that are observed by the dynamics of the underlying PN. In the context of the general PN theory, there are two types of invariants: the p and t-invariants. These two concepts are formally defined as follows:

Definition 16 *A* p-invariant *of a PN* $\mathcal{N} = \langle P, T, W, \mathbf{m}_0 \rangle$ *is a* $|P|$*-dimensional vector* $\mathbf{y}$ *satisfying* $\mathbf{y}^T \cdot \Theta = 0$. *On the other hand, a* t-invariant *of* $\mathcal{N}$ *is a* $|T|$*-dimensional vector* $\mathbf{x}$ *satisfying* $\Theta \cdot \mathbf{x} = 0$. *Furthermore, a* p*-invariant (resp.,* t*-invariant) with nonnegative elements is further characterized as a* p*(resp.,* t*)*-semiflow.

In the light of Equation A.2, the invariance property expressed by a p-invariant $\mathbf{y}$ is that $\mathbf{y}^T \cdot \mathbf{m} = \mathbf{y}^T \cdot \mathbf{m}_0$, for all $\mathbf{m} \in R(\mathcal{N}, \mathbf{m}_0)$. Similarly, Equation A.2 implies that for any t-invariant $\mathbf{x}$, $\mathbf{m} = \mathbf{m}_0 + \Theta \cdot \mathbf{x} = \mathbf{m}_0$.

Given a p-invariant $\mathbf{y}$, its *support* is defined as $\|\mathbf{y}\| = \{p \in P \mid \mathbf{y}[p] \neq 0\}$. A p-invariant $\mathbf{y}$ is said to be *minimal iff* there is no p-invariant $\mathbf{y}'$ s.t. $\|\mathbf{y}'\| \subset \|\mathbf{y}\|$, and its nonzero components are relatively prime. Furthermore, similar definitions apply to the case of a t-invariant $\mathbf{x}$.

[1] The reader should notice that the concept of "deadlock" in the PN modeling framework does not have the same meaning with the usage of this term in the context of Definition 5 in Chapter 2. More specifically, in the context of Definition 5, and in most of the main text of this manuscript, "deadlock" implies a "resource allocation deadlock", and this usage of the term is motivated by its similar employment in the corresponding theory of complex RAS [59], that has its own strong presence in the broader DES literature. Whenever there is a need to clarify / disambiguate this term in the presented developments, we shall differentiate between a "PN deadlock" and a "RAS deadlock".

A PN $\mathcal{N}$ is *conservative iff* there exists a p-semiflow $\mathbf{y}$ with all its components strictly greater than zero. Obviously, any conservative PN $\mathcal{N}$ is bounded. Also, a PN $\mathcal{N}$ is *consistent iff* there exists a t-semiflow $\mathbf{x}$ with all its components strictly greater than zero. Finally, we also have the following definition:

Definition 17 *A PN $\mathcal{N}$ is a* mono-t-semiflow net *iff it is consistent, conservative, and has only one minimal t-semiflow.*

REFERENCES

[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modeling with Generalized Stochastic Petri Nets*. John Wiley & Sons, 1994.

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

[3] S. Asmussen and P. W. Glynn, *Stochastic Simulation: Algorithms and Analysis*. NY, NY: Springer, 2007.

[4] R. Bellman, *Applied Dynamic Programming*. Princeton, N. J.: Princeton University, 1957.

[5] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vol. 2 (4th ed.)* Belmont, MA: Athena Scientific, 2012.

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

[7] D. Bertsimas, E. Nasrabadi, and I. C. Paschalidis, "Robust fluid processing networks," *IEEE Trans. on Automatic Control*, vol. 60, pp. 715–728, 2015.

[8] X. Cao, *Stochastic Learning and Optimization*. NY,NY: Springer, 2007.

[9] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems (2nd ed.)* NY, NY: Springer, 2008.

[10] H. Chen and D. D. Yao, *Fundamentals of Queueing Networks*. NY, NY: Springer, 2001.

[11] Y. F. Chen and Z. W. Li, "Design of a maximally permissive liveness-enforcing supervisor with a compressed supervisory structure for flexible manufacturing systems," *Automatica*, vol. 47, pp. 1028–1034, 2011.

[12] J. Y. Choi and S. A. Reveliotis, "A generalized stochastic Petri net model for performance analysis and control of capacitated re-entrant lines," *IEEE Trans. on Robotics and Automation*, vol. 19, pp. 474–480, 2003.

[13] ——, "Relative value function approximation for the capacitated re-entrant line scheduling problem," *IEEE Trans. on Automation Science and Engineering*, vol. 2, pp. 285–299, 2005.

[14] W. L. Cooper, S. G. Henderson, and M. E. Lewis, "Convergence of simulation-based policy iteration," *Probability in Engineering and Information Science*, vol. 17, pp. 213–234, 2003.

[15] R. Cordone, A. Nazeem, L. Piroddi, and S. Reveliotis, "Designing optimal deadlock avoidance policies for sequential resource allocation systems through classification theory: Existence results and customized algorithms," *IEEE Trans. Autom. Control*, vol. 58, pp. 2772–2787, 2013.

[16] J. G. Dai, "On positive Harris recurrence of multiclass queueing networks: A unified approach via fluid limit models," *Annals of Applied Probability*, vol. 5, pp. 49–77, 1995.

[17] ——, "Stablity of fluid and stochastic processing networks," Dept. of Mathematical Sciences, University of Aarhus, Denmark, Tech. Rep. Miscellanea, No. 9, 1998.

[18] J. G. Dai and W. Lin, "Maximum Pressure Policies in Stochastic Processing Networks," *Operations Research*, vol. 53, pp. 197–218, 2005.

[19] J. G. Dai and J. H. Vande Vate, "The stability od two-station multitype fluid networks," *Operations Reseach*, vol. 48, pp. 721–744, 2000.

[20] J. G. Dai and G. Weiss, "Stability and instability of fluid models for reentrant lines," *Mathematics of Operations Research*, vol. 21, pp. 115–134, 1996.

[21] J. Ezpeleta, J. M. Colom, and J. Martinez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Trans. on R&A*, vol. 11, pp. 173–184, 1995.

[22] Z. Fei, S. Reveliotis, S. Miremadi, and K. Akesson, "A BDD-based approach for designing maximally permissive deadlock avoidance policies for complex resource allocation systems," *IEEE Trans. on Automation Science and Engineering*, vol. 12, pp. 990–1006, 2015.

[23] A. Giua, F. DiCesare, and M. Silva, "Generalized mutual exclusion constraints on nets with uncontrollable transitions," in *Proceedings of the 1992 IEEE Intl. Conference on Systems, Man and Cybernetics*, IEEE, Chicago, IL, 1992, pp. 974–979.

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

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

[26] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*. Reading, MA: Addison-Wesley, 1979.

[27] H. Hu, M. Zhou, and Z. Li, "Liveness and ratio-enforcing supervision of automated manufacturing systems using Petri nets," *IEEE Trans. on Systems, Man and Cybernetics – Part A: Systems and Humans*, vol. 42, pp. 392–403, 2012.

[28] M. V. Iordache and P. J. Antsaklis, "Design of t-liveness enforcing supervisors in petri nets," *IEEE Trans. on Automatic Control*, vol. 48, pp. 1962–1974, 2003.

[29] ——, *Supervisory Control of Concurrent Systems: A Petri net structural approach*. Boston, MA: Birkhäuser, 2006.

[30] S.-H. Kim, "Comparison with a standard via fully sequential procedures," *ACM Trans. on Modeling and Computer Simulation*, vol. 15, pp. 155–174, 205.

[31] S.-H. Kim and B. L. Nelson, "Selecting the best system," in *Handbook in Operations Research and Management Science: Simulation*, S. G. Henderson and B. L. Nelson, Eds., Elsevier, 2006.

[32] S.-H. Kim, "Comparison with a standard via fully sequential procedures," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 15, no. 2, pp. 155–174, 2005.

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

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

[35] P. R. Kumar and S. P. Meyn, "Duality and linear programs for stability and performance analysis of queueing networks and scheduling policies," *IEEE Trans. Autom. Control*, vol. 41, pp. 4–17, 1996.

[36] H. J. Kushner and G. G. Yin, *Stochastic Approximation and Recursive Algorithms and Applications*. NY, NY: Springer, 2003.

[37] M. Lawley, S. Reveliotis, and P. Ferreira, "A correct and scalable deadlock avoidance policy for flexible manufacturing systems," *IEEE Trans. on Robotics & Automation*, vol. 14, pp. 796–809, 1998.

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

[39] R. Li, "Performance optimization of complex resource allocation systems," PhD thesis, ISyE, Georgia Tech, Atlanta, GA, 2016.

[40] R. Li and S. Reveliotis, "Designing parsimonious scheduling policies for complex resource allocation systems through concurrency theory," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 26, pp. 511–537, 2016.

[41] ——, "Performance optimization for a class of generalized stochastic Petri nets," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 25, pp. 387–417, 2015.

[42] Z. Li and M. Zhou, "Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 34, no. 1, pp. 38–51, 2004.

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

[44] C. Mahulea, "Timed continuous petri nets:quantitative analysis, observability and control," PhD thesis, Universidad de Zaragoza, Zaragoza, Spain, 2007.

[45] D. C. Marinescu, *Internet-Based Workflow Management: Towards a Semantic Web*. NY,NY: Wiley Interscience, 2002.

[46] S. Meyn, *Control Techniques for Complex Networks*. Cambridge, UK: Cambridge University Press, 2008.

[47] S. P. Meyn, "Stability and optimization of multi-class queueing networks and their fluid models," *Lectures in Applied Mathematics*, vol. 33, pp. 175–199, 1997.

[48] J. O. Moody and P. J. Antsaklis, *Supervisory Control of Discrete Event Systems using Petri nets*. Boston, MA: Kluwer Academic Pub., 1998.

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

[50] A. Nazeem and S. Reveliotis, "A practical approach for maximally permissive liveness-enforcing supervision of complex resource allocation systems," *IEEE Trans. on Automation Science and Engineering*, vol. 8, pp. 766–779, 2011.

[51] ——, "Designing maximally permissive deadlock avoidance policies for sequential resource allocation systems through classification theory: The non-linear case," *IEEE Trans. on Automatic Control*, vol. 57, pp. 1670–1684, 2012.

[52] ——, "Efficient enumeration of minimal unsafe states in complex resource allocation systems," *IEEE Trans. on Automation Science & Engineering*, vol. 11, pp. 111–124, 2014.

[53] A. Nazeem, S. Reveliotis, Y. Wang, and S. Lafortune, "Optimal deadlock avoidance for complex resource allocation systems through classification theory," in *Proceedings of the 10th Intl. Workshop on Discrete Event Systems*, IFAC, Berlin, Germany, 2010, pp. 277–284.

[54] B. Nelson and D. Goldsman, "Comparison with a standard in simulation experiments," *Management Science*, vol. 47, pp. 449–463, 2001.

[55] J. Park and S. A. Reveliotis, "Deadlock avoidance in sequential resource allocation systems with multiple resource acquisitions and flexible routings," *IEEE Trans. on Automatic Control*, vol. 46, pp. 1572–1583, 2001.

[56] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. NY, NY: Wiley, 2007.

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

[58] L. Recalde, E. Teruel, and M Sliva, "Autonomous continuous P/T systems," in *Applications and Theory of Perti Nets 1999*, Williamsburg, VA, 1999, pp. 107–126.

[59] S. Reveliotis, "Logical Control of Complex Resource Allocation Systems," *NOW Series on Foundations and Trends in Systems and Control*, vol. 4, pp. 1–224, 2017.

[60] S. Reveliotis and A. Nazeem, "Optimal linear separation of the safe and unsafe subspaces of sequential RAS as a set-covering problem: Algorithmic procedures and geometric insights," *SIAM Journal on Control and Optimization*, vol. 51, pp. 1707–1726, 2013.

[61] S. Reveliotis and E. Roszkowska, "On the complexity of maximally permissive deadlock avoidance in multi-vehicle traffic systems," *IEEE Trans. on Automatic Control*, vol. 55, pp. 1646–1651, 2010.

[62] S. A. Reveliotis, "On the siphon-based characterization of liveness in sequential resource allocation systems," in *Applications and Theory of Perti Nets 2003*, Eindhoven, NL, 2003, pp. 241–255.

[63] ——, *Real-time Management of Resource Allocation Systems: A Discrete Event Systems Approach*. NY, NY: Springer, 2005.

[64] ——, "The destabilizing effect of blocking due to finite buffering capacity in multiclass queueing networks," *IEEE Trans. on Autom. Control*, vol. 45, pp. 585–588, 2000.

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

[66] S. M. Ross, *A First Course in Probability (9th edition)*. N.Y.: Pearson, 2014.

[67] ——, *Stochastic Processes*. N.Y.: Wiley, 1983.

[68] P. Rowe, "The paired t-test," *Essential Statistics for the Pharmaceutical Sciences, Second Edition*, pp. 163–175,

[69] M. Silva, E. Teruel, and J. M. Colom, "Linear algebraic and linear programming techniques for the analysis of place/transition net systems," in *Lecture Notes in Computer Science, Vol. 1491*, W. Reisig and G. Rozenberg, Eds., Springer-Verlag, 1998, pp. 309–373.

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

[71] F. Tricas, F. Garcia-Valles, J. M. Colom, and J. Ezpeleta, "A Petri net structure-based deadlock prevention solution for sequential resource allocation systems," in *Proceedings of the ICRA 2005*, IEEE, Barcelona, 2005, pp. 271–277.

[72] L. M. Wein, "Scheduling semiconductor wafer fabrication," *IEEE Transactions on semiconductor manufacturing*, vol. 1, no. 3, pp. 115–130, 1988.

[73] G. Weiss, "On optimal draining of re-entrant fluid lines," *IMA volumes in mathematics and its applications*, vol. 71, pp. 91–91, 1995.

[74] ——, "Scheduling and control of manufacturing systems," in *Proceedings of the 37th Allerton Conference*, University of Illinois, Allerton, IL, 2000, pp. –.

[75] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[76] W. M. Wonham, "Supervisory control of discrete event systems," Electrical & Computer Eng., University of Toronto, Tech. Rep. ECE 1636F / 1637S 2013-14, 2006.

VITA

Michael Ibrahim was born in Cairo, Egypt. He received a Bachelor's Degree in Computer Engineering from Cairo University in 2012, and a Master's degree in Computer Engineering from Cairo University in 2015. Since 2015, he has been pursuing a Ph.D. degree in Industrial Engineering at the Georgia Institute of Technology. His research interests include discrete event systems, operations research, and machine learning.