PERFORMANCE OPTIMIZATION OF COMPLEX RESOURCE ALLOCATION SYSTEMS

A Dissertation
Presented to
The Academic Faculty

By

Ran Li

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

Georgia Institute of Technology
December 2016

PERFORMANCE OPTIMIZATION OF
COMPLEX RESOURCE ALLOCATION SYSTEMS

Approved by:

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

Dr. Enlu Zhou
School of Industrial and Systems Engineering
Georgia Institute of Technology

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

Dr. Ioannis Ch. Paschalidis
Department of Electrical and Computer Engineering
Boston University

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

Date Approved: September 12, 2016

To my parents and my wife,

for their love and selfless support.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my advisor, Professor Spyros Reveliotis, for his help, guidance and support during my studies in Georgia Tech. It was one of the most interesting parts of my Ph.D. life to learn, discuss, develop and compare ways to solve various problems arising in the research project, together with Spyros. His thoughtful ideas and thorough guidance helped me a lot in enhancing my skills and knowledge as a researcher. His passion and rigorous attitude to research set himself an example for me. I could not imagine my research without his advising.

Professor Hayriye Ayhan, Professor Ioannis Ch. Paschalidis, Professor Kamran Paynabar and Professor Enlu Zhou, thank you for serving on my thesis committee and providing valuable suggestions. Professor Ron Johnson and Professor Chen Zhou, thank you for teaching me the way of teaching. Professor Jim Dai, Professor Santanu Dey, Professor Jan Shi, Professor Magnus Egerstedt, and many other faculties and staffs in ISyE and other departments of Georgia Tech, thank you for your top-level classes, seminars and services.

I want to thank Dr. Hongwei Liao and other colleagues from General Electric, for their help and guidance when I was an intern student there. I have learned a lot on the practical side of operations research that could never have been experienced on campus. And the life in Schenectady was also delightful!

Life would be dull and more difficult without my wonderful fellow graduate students in Georgia Tech. I would like to thank Yanling Chang, Qiushi Chen, Carl Morris, Jan Valchy, Xinchang Wang, Chenxi Zeng, Chengliang Zhang, and many others. I really enjoyed the time spent with you on the discussion of the hard homework and

research problems, the sharing of the job-hunting information, and the relaxation in the dining halls, the gym and the state parks. I am so lucky to meet you, guys, in my life!

I am deeply grateful to my wife, Chenxing, who has always been with me and taken care of our home in Atlanta. I would like to thank my father's cousin, Frank Sun, and his family in Atlanta, for helping me settle down everything when I first came to this beautiful and vibrant city. I also want to express my gratitude to my extended family members in Beijing: my parents, parents-in-law, grandparents, uncles, aunts, and cousins, for their endless support, care and encouragement, regardless of the great geographical distance.

Finally, I acknowledge the NSF support of this research and the Morris fellowship which additionally stipended my tuition and living expenses in my first year.

TABLE OF CONTENTS

DEDICATION . . . . . **iii**

ACKNOWLEDGEMENTS . . . . . **iv**

LIST OF TABLES . . . . . **ix**

LIST OF FIGURES . . . . . **x**

SUMMARY . . . . . **xii**

I INTRODUCTION **1**

II BACKGROUND **13**

2.1 Formal definition of the sequential resource allocation system 13

2.2 The RAS deadlock avoidance problem 17

2.2.1 A classifier-based method for deadlock avoidance 20

2.3 The GSPN reward model and its performance evaluation 23

2.3.1 Definition and basic properties of Petri nets 23

2.3.2 The extension of a PN to GSPN 26

2.3.3 GSPN reward models and their performance evaluation . . . 29

2.4 The capacitated re-entrant line, its RAS-based representation, and the CRL scheduling problem of throughput maximization 33

III FORMAL MODELING OF THE TIMED RAS DYNAMICS AS A GSPN, AND THE CORRESPONDING PERFORMANCE CONTROL PROBLEM **40**

3.1 GSPN-based modeling of the timed behavior of the RAS 40

3.1.1 Control elements for the RAS-modeling GSPN 46

3.1.2 Some additional features of the RAS-modeling GSPN 50

3.2 Specialization of the GSPN-based modeling of the RAS performance control problem to the CRL scheduling problem 53

3.3 The performance optimization of the RAS-modeling GSPN and its mathematical programming formulation 60

3.4 Complexity considerations 66

3.5 A closing remark 68

IV CONTROLLING THE REPRESENTATIONAL COMPLEXITY 71

4.1 A random-switch refinement process 71

4.1.1 Formalization of the random-switch refinement process 73

4.1.2 Construction of a "minimal" refined policy space 74

4.1.3 Some sufficient conditions on $\mathcal{E}_u^{\hat{\Pi}_2}(M)$ that maintain the corresponding untimed tangible reach 85

4.1.4 Specialization to the CRL case 88

4.1.5 A numerical experiment for the random-switch refinement processes developed in this chapter 97

4.1.6 The policy space $\hat{\Pi}_3$ 100

4.2 Static random switches 100

4.2.1 Applying static random switches 101

4.2.2 A numerical experiment for the application of static random switches 103

4.2.3 The potential performance losses incurred by the employment of static random switches and a plan for their retrieval 104

4.3 An alternative representation for random switches that facilitates partial disaggregation 105

V SOLVING THE FORMULATED SCHEDULING PROBLEM WITH STOCHASTIC APPROXIMATION 109

5.1 The basic stochastic approximation framework 110

5.1.1 An asymptotic convergence result for the considered SA algorithm 113

5.2 Customizing the basic SA algorithm to the considered performance optimization problem 115

5.2.1 The considered optimization problem 115

5.2.2 Determining the improvement direction 118

5.2.3 Specification of the remaining ingredients of the SA recursion 130

5.2.4 An asymptotically convergent SA algorithm for the considered optimization problem 131

5.2.5 Some additional considerations 134

5.3 A more practical implementation of the SA algorithm 136

5.3.1 A "baseline" experiment 137

5.3.2 Sample-size control in the estimation of the improvement direction 142

5.3.3 A statistical test for the terminating condition 160

5.3.4 The practical version of the proposed SA algorithm 170

5.4 Extensions for partially disaggregated random switches 175

5.4.1 Modifying the SA algorithm to cope with partial disaggregation 175

5.4.2 Demonstrating the improvement potential of partial disaggregation 178

5.4.3 Some further observations on the experiment of Section 5.4.2 181

VI A SUMMARY OF THE MAJOR CONTRIBUTIONS AND POSSIBLE FUTURE EXTENSIONS 186

6.1 The major contributions 186

6.2 Possible future extensions 188

APPENDIX A — A MARKOV DECISION PROCESS FOR THE CONSIDERED PERFORMANCE OPTIMIZATION PROBLEM 192

APPENDIX B — SUPPLEMENTARY DATA 200

APPENDIX C — AN EXAMPLE FOR THE VIOLATION OF THE Π_2-IRREDUCIBILITY OF THE REFINED TRANSITION SETS RETURNED BY ALGORITHM 3 203

APPENDIX D — AN IMPLEMENTATION OF THE PROJECTION OPERATOR EMPLOYED IN THE ALGORITHMS OF CHAPTER 5 206

REFERENCES 215

VITA 223

LIST OF TABLES

2.1 A RAS taxonomy [86] 17

3.1 The operational semantics encoded by the GSPN in Figure 3.5 59

4.1 The CRL configurations employed in the numerical experiment that is reported in Sections 4.1.5 and 4.2.2 [60]. 98

4.2 Comparison of the numbers of random switches (R.S.) and decision variables (D.V.) for the CRL configurations of Table 4.1 that result from (i) no refinement, (ii) Algorithm 2 (general refinement), and (iii) Algorithm 3 (CRL-customized refinement) 99

4.3 Comparison of the numbers of random switches (R.S.) and decision variables (D.V.) for the CRL configurations of Table 4.1 when applying the static random switches 104

5.1 The values of the objective function at the solutions obtained in the baseline experiment for the first 16 CRL configurations of Table 4.1 . 139

5.2 Comparison of the coefficients of variation for the three sample-size control methods on the first 16 CRL configurations of Table 4.1 . . . 156

5.3 Comparison of the growth of the N_{new} estimates between the two policies $\bar{\zeta}_0$ and $\bar{\zeta}_{500}$ for the three sample-size control methods on the first 16 CRL configurations of Table 4.1 157

5.4 The values of the objective function at the solution points returned by Algorithm 6 for the first 16 CRL configurations of Table 4.1 173

5.5 The values at the 500-th SA iteration of the decision variables employed by the partially disaggregated random switch of $\mathcal{E}_3$ 182

B.1 Markings enumerated from the GSPN in Example 1 200

B.2 The sample averages and standard deviations for N_{new} estimates under the Shapiro-Mello control method 201

B.3 The sample averages and standard deviations for N_{new} estimates under the control method that restricts the volume of the confidence region 202

B.4 The sample averages and standard deviations for N_{new} estimates under the control method that restricts the maximum length of the confidence intervals 202

LIST OF FIGURES

1.1 An event-driven RAS control framework [86] 3

1.2 The methodological framework 9

2.1 An example PN and its conversion to a GSPN 27

2.2 The CRL model of Example 1 37

3.1 The sequential logic for the RAS process type considered in Example 2, and its PN model 44

3.2 Introduction of the timed behavior in Example 2 46

3.3 The set of transitions that "offset" the changes on the marking M that are incurred by the firing of transition t_1 52

3.4 The GSPN model of a CRL stage J_j, before and after simplification . 54

3.5 The GSPN model for the CRL scheduling problem in Example 1 . . . 58

3.6 The Π_2-conditional state transition diagram of the underlying semi-Markov process for Example 1 65

3.7 An illustration of the net structure that integrates the action of deliberate idleness in the RAS-modeling GSPN 69

4.1 The marking M_1 of Example 3 82

4.2 The local state transition diagram of $\mathcal{UR}^{\Pi_2}(M_1)$ in Example 3 and the sets of untimed tangible reaches of the vanishing nodes 83

4.3 The local STD of $\mathcal{UR}^{\hat{\Pi}_2}(M_1)$ in Example 3 under Algorithm 3 95

5.1 Comparison of the throughput improvements made through the first 100 and 500 iterations of Algorithm 5 for the 16 CRL configurations . 141

5.2 A generic scheme for sample-size control in the computation of the sample average Y_{A} 143

5.3 The gradient estimate $\widehat{\nabla\eta}$, the true gradient vector $\nabla\eta$, and the desired confidence region (adapted from [44]) 148

5.4 Comparison of the confidence regions specified by the Shapiro-Mello and the variance-based methods for different norms of the gradient estimates. 159

5.5 Comparison of the throughput improvements obtained from the basic and the enhanced SA algorithms for the 16 CRL configurations of Table 4.1 174

5.6 The reductions with respect to the total number of transitions and the computing time incurred by the presented algorithmic enhancements during the algorithm application on the 16 CRL configurations of Table 4.1 174

5.7 The performance improvements attained for the CRL configuration that is considered in the performed experiment, through the partial disaggregation of the static random switches of the original policy space $\hat{\Pi}_3^S$ 180

A.1 The Π_2-conditional state transition diagram of the underlying semi-Markov process for Example 1 with a highlighted region on the local STD related to the construction of the example pre-decision state $s_{2,6,7'}$ 196

A.2 The Π_2-conditional state transition diagram of the Markov decision process for Example 1 197

C.1 The application of Propositions 7 and 8 on the untimed reach of the marking M_{V1} in the example that is considered in this appendix. . . . 204

SUMMARY

The typical control objective for a sequential resource allocation system (RAS) is the optimization of some (time-based) performance index, while ensuring the logical/behavioral correctness of the underlying automated processes. There is a rich set of results on logical control, and these results are quite effective in their ability to control RAS with very complex structure and behavior. On the other hand, the existing results on the performance-oriented control, i.e., the scheduling, of RAS are limited. The research program presented in this document seeks to provide a complete and systematic methodological framework for the RAS scheduling problem with the integration of the logical control, by leveraging the formal representations of the RAS behavior. These representations enable (i) the formulation of the RAS scheduling problem in a way that takes into consideration the RAS behavioral control requirements, and (ii) the definition of pertinent policy spaces that provide an effective trade-off between the representational and computational complexity of the pursued formulations and the operational efficiency of the derived policies. Although the presented methodological framework can be applied to any general RAS, this research mainly focuses on a class of RAS that abstracts the capacitated re-entrant line (CRL) model, and it uses this RAS class to demonstrate the overall methodology.

The presented framework is divided into two parts: a "modeling" and an "algorithm" part. The "modeling" part consists of the procedures to model the RAS dynamics as a generalized stochastic Petri net (GSPN), that supports a seamless integration of, both, the logical and performance control problems. This part also formulates the scheduling problem of the performance optimization of a logically controlled

GSPN as a mathematical programming (MP) problem that is derived from the semi-Markov process (SMP) modeling the timed dynamics of this GSPN. In the resulting MP formulation, the decision variables are parameters that can adjust the embedded transition probabilities of the SMP, and the objective function is the steady-state average reward with respect to a given immediate reward function.

The problem of the explosive size of the MP formulation with respect to the size of the underlying RAS is addressed by (re-)defining the target policy space and its detailed representation. More specifically, three steps of complexity control are applied on the original policy space: The first step is a "refinement" process that simplifies the representation of the original policy space but does not harm its performance potential. The second step is a "restriction", which further reduces the number of decision variables by coupling the decision-making logic that corresponds to "similar" states. Numerical studies show a dramatic reduction on the dimension of the solution space with the implementation of the first two steps. The third step of the proposed complexity control method is a partial "disaggregation" process that tries to break certain couplings formed in the second step, and thus obtain more degrees of freedom to pursue a further improvement on the optimized system performance under the applied aggregation. This third step is the mechanism that explicitly controls the trade-off between the representational and computational complexity of the target policies and their operational efficiency.

Since the complexity control that is adopted in the "modeling" part is applied only on the policy space, the analytical solution of the resulting MP is still intractable because the evaluation of the objective function requires the underlying steady-state distribution of the system sojourn at each state. As a consequence, this MP is solved through a simulation optimization method called stochastic approximation (SA) in the "algorithm" part. To this end, the adopted GSPN representation has provided a

succinct and efficient simulation platform, and it has facilitated the systematic estimation of the necessary gradients. At the same time, the adopted SA algorithms have been strengthened by the integration in their basic evaluation and exploration logic of results coming from the area of statistical inference. These results have enabled our SA algorithms to proceed to a near-optimal region in a robust and stable way, while avoiding the expenditure of computational resources in areas of the underlying response surface with little potential gain.

CHAPTER I

INTRODUCTION

Complex resource allocation takes place in the operational context of various contemporary technological applications. These applications range from manufacturing to transportation and IT systems. For instance, in a 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 "competitions" 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.

Complex resource allocation in a manufacturing system may also happen beyond the scope of the fabrication process itself. An example can be seen in [83]: in the domain of material handling systems (MHS), such as the zone-controlled automated guided vehicle (AGV) system, the links of the guide path network are split into a number of segments called "zones". To avoid possible collisions among the traveling vehicles, at any time each zone can be occupied by at most one vehicle. Meanwhile, the zones must be allocated to the vehicles in an efficient way so that some performance-oriented goals are achieved; for instance, each vehicle can finish its tasks in a timely manner or the supported transfer rate is maximized. Furthermore, the

above application in the MHS domain can be extended to more general traffic systems involving the circulation of a number of autonomous agents in a confined area [78, 91].

In the IT area of multithread computing [64, 33, 34], a number of simultaneously running computing tasks – or the "threads" of a computer program – require access to some common resources such as memory registers and/or data files. Accessing a certain resource by a thread must be negotiated with the system controller to make sure that the program execution is correct and efficient.

In summary, all the aforementioned applications share these common characteristics: (i) there exist a set of reusable but limited *resources*, such as the machines, the guide path zones, the memory registers and the data files; (ii) there exist a set of *processes*, such as the parts, the vehicles and the program threads, 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, the moving of the vehicles, and the computing of the program threads; (iii) the tasks of each process should be performed in a *sequential* manner, such as the "specified orders of the machines" for the parts, the routes of the vehicles, and the organization of the code behind the program threads; finally (iv) the resource requests from different tasks may overlap, but the shared requested resources are not sufficient, and in such a case, an *arbitration* is needed to assign *priorities* to the different tasks competing for the insufficient resources. Typically, the objective for such arbitrations is the *optimization* of some (time-based) *performance index*, like the maximization of the production rate of the aforementioned manufacturing cell, and the control of the travel and computing times in the other two applications.

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

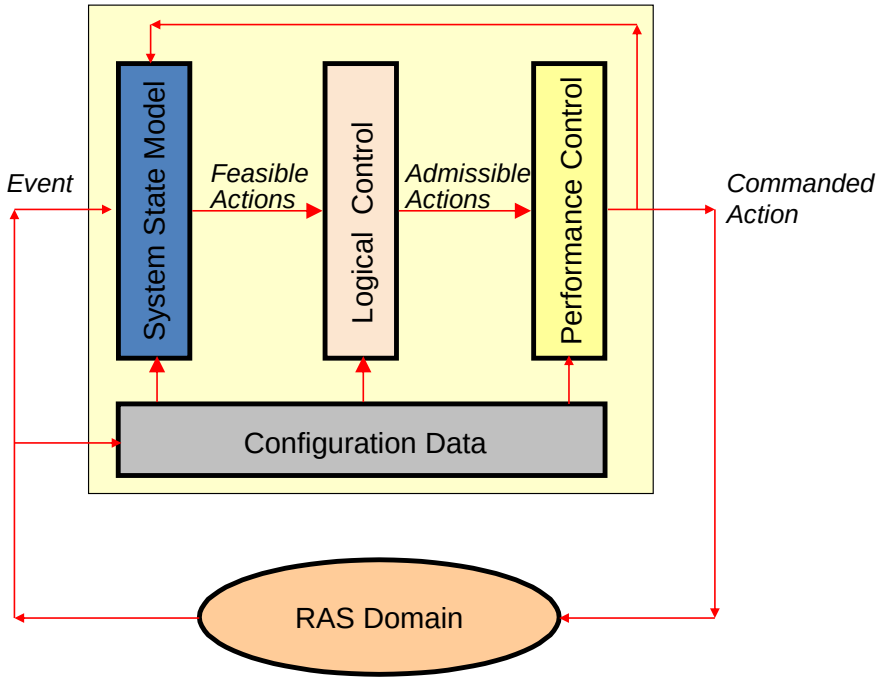


Figure 1.1: An event-driven RAS control framework [86]

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 an effort to address both control objectives of logical correctness and performance optimization for the complex resource allocation functions that were discussed in the previous paragraphs, the relevant research community has introduced the formal abstraction of the *sequential Resource Allocation System (RAS)* [86]. Furthermore, the RAS dynamics have been modeled and analyzed through a set of modeling frameworks and techniques provided by the Discrete Event System (DES) theory [16]. More specifically, Figure 1.1 gives an integrated event-driven control framework for the considered RAS model. The controller responds to the events taking place in the controlled RAS by maintaining a representation of the system *state*. This state

representation allows the controller to determine the set of *feasible actions*. From these feasible actions, the logical controller filters out the set of *admissible actions*, to prevent the system from reaching any logically incorrect (i.e., deadlock) states. Finally, the performance controller selects an admissible action to be commanded on the system for performance optimization. This selection essentially defines a *scheduling* problem.

During the past two decades, a rich set of results on logical control has become available, and these results are quite effective in their ability to control RAS with very complex structure and behavior [102, 89, 87]. On the other hand, the existing results on the performance-oriented control of RAS are limited. Especially, there is a lack of a complete and systematic methodology for the RAS scheduling problem with the integration of the logical control. The research program presented in this document seeks to provide such a complete and systematic methodological framework for the RAS scheduling problem, by leveraging the formal DES-based representations of the RAS behavior. These representations will enable (i) the formulation of the RAS scheduling problem in a way that takes into consideration the RAS behavioral control requirements, and (ii) the definition of pertinent policy spaces that provide an effective trade-off between the representational and computational complexity of the pursued formulations and the operational efficiency of the derived policies.

More specifically, the tool for the representation of the RAS dynamics that is employed in this work is the generalized stochastic Petri net (GSPN). The GSPN is an extension of the Petri net (PN), a widely applied modeling and analysis tool [73], especially in the domain of DES [16]. It is well known that a PN can model the qualitative behavior of a RAS and the necessary logical control policies [5, 29, 79, 62]. In addition, by introducing timing features to the dynamics of the PN, a GSPN can model the timed dynamics of a RAS. As a result, in a GSPN, the typical performance measures of a RAS (such as process throughputs, delays and concentrations) can

be modeled and analyzed in a way that is also considering the qualitative aspects of the RAS behavior and the corresponding logical control policies; i.e., the GSPN representation of a RAS is able to support a seamless integration of, both, the logical and performance control problems.

Furthermore, the considered GSPN models inherit the ability of the basic PN model to express additional requirements for the behavior / operational logic of the underlying system, such as "fairness" requirements with respect to the various processes or throughput-ratio constraints for a production line with more than one products. One can implement these additional requirements with appropriate augmentation of the net structure.

The performance optimization of the developed GSPN models, in principle, can be based on a set of classical methods borrowed from the theory of Markov decision processes (MDPs) [82]. More specifically, the performance of a GSPN can be analyzed through a Markovian model [2]. And this performance evaluation method enables the eventual formulation of the performance optimization problem of the considered GSPNs as an average-reward Markov decision process (AR-MDP) whose objective is the maximization of the steady-state average reward. In fact, an AR-MDP formulation of the considered scheduling problem of the logically-controlled RAS can also be derived more directly from the underlying RAS model [86]. In either way, it is essentially an AR-MDP model that is applied to the selection of "admissible actions" in the event-driven control scheme of Figure 1.1. Classical AR-MDP approaches, such as linear programming, value iteration or policy iteration, will specify an optimal solution for the scheduling problem considered in Figure 1.1 in the form of a look-up table that pairs the RAS decision states with action choices [82].[1] Unfortunately, all these approaches enumerate explicitly the underlying state space, and therefore,

[1]More details for building the "communicating" AR-MDP model for the performance optimization of a RAS can be found in Appendix A.

suffer from the "curse of dimensionality" [8]. Indeed, the size of the state space of the considered AR-MDPs may be intractable even for a moderately sized RAS.[2] Furthermore, the representation of an optimal policy itself ends up being a hard task since a look-up table representation of this policy requires the specification of the selected action at each state.

A number of approaches have been developed to solve MDPs with large state spaces. These approaches try to approximate some aspects of the large state space MDPs so that the representational and computational complexity for their solution can be significantly reduced. In particular, the explicit enumeration of the state space should be avoided and the solution should be represented in a more parsimonious way than a look-up table. But the attained complexity reduction often comes at the cost of a deterioration in performance, or a reduction of the operational efficiency. Hence, there is a trade-off between complexity and optimality.

The aforementioned methods are collectively known as approximate dynamic programming (ADP) [11, 81]. The most common ADP methods rely on the approximation of the (relative) value of the various states, which can be seen as the potential or expected benefit of being at the considered state under operation by the optimal policy.[3] And the optimal policy given by these ADP methods is a "greedy" choice of the admissible action in a way that maximizes a certain functional of the relative value function. The relative value function itself can be approximated by taking linear combinations of some pre-defined feature functions of the states; the coefficients

[2]In fact, the state space of the considered AR-MDPs is a composition, thus a Cartesian product, of the state spaces of the various interacting components in the system. Hence, if the system contains n components and each component has a state space of size m, then the size of the state space of the entire system is $O(m^n)$.

[3]In a finite horizon or discounted MDP, the value function typically reflects the expected total reward in the future for a given state. But in an AR-MDP, this value is infinite due to absence of the discount factor. Thus, the notion of "relative value function" is used instead. A relative value function contrasts the original values of the different states; in particular, the relative value between any two given states is the difference of the accumulated reward when starting the stochastic process from the corresponding states. In this document, the optimization problem is an AR-MDP, so only relative value functions are used in the sequel.

of these linear combinations can be tuned through some simulation-based learning algorithms, such as temporal difference (TD), that make the corresponding Poisson equations hold in an approximate sense. This kind of value function approximation is also known as a version of Neuro-Dynamic Programming (NDP) [13]. In [21] such a method is applied to solve a scheduling problem in a capacitated re-entrant line (CRL), which can be seen as a special type of RAS. But typically, the definition of a good set of feature functions remains a hard problem and is usually relatively ad-hoc. In addition, in the above ADP methods, the trade-off between complexity and optimality is not explicitly controlled.

Therefore, this research program takes a different perspective than the ADP approaches discussed in the previous paragraph. More specifically, the considered scheduling problem is formulated as a mathematical programming (MP) problem that is derived from the semi-Markov process (SMP) modeling the timed dynamics of the logically controlled GSPN, and the association of certain states of this SMP with the reward rates that model the performance elements of interest. For instance, the states that involve part completion and unloading in the SMP modeling the operations of a manufacturing cell can be associated with some positive reward if the objective is the maximization of the cell throughput. In the resulting MP formulation, the decision variables are parameters that can adjust the underlying performance control policy, and the objective function is the steady-state average reward of the SMP. However, similar to the MDP model, the size of the MP formulation, in terms of, both, the numbers of the decision variables and the constraints, is still super-polynomial with respect to the size of the underlying RAS.[4] This problem is addressed by (re-)defining the target policy space and its detailed representation. More specifically, three steps of complexity control are applied on the original policy space: The first

[4]Generally speaking, the RAS size is the size of any parsimonious representation of its structure. More technical definitions of all these concepts are provided in later parts of this document.

step is a "refinement" process that simplifies the representation of the original policy space but does not harm its performance potential; i.e., the policy space after this refinement contains at least one optimal policy from the original policy space. The second step is a "restriction", which further reduces the number of decision variables by coupling the decision-making logic that corresponds to "similar" states. From an operational standpoint, the introduced restriction will keep all the "static-priority" policies that are popular in the industrial practice. From the perspective of MDP and ADP theory, this second step constitutes an approximation of the policy space through the imposition of an aggregation scheme on the underlying state space, and it can be seen as a special type of ADP. The third step of the proposed complexity control method is a partial "disaggregation" process that tries to break certain couplings formed in the second step, and thus obtain more degrees of freedom to pursue a further improvement on the optimized system performance under the applied aggregation. This third step is the mechanism that explicitly controls the trade-off between the representational and computational complexity of the target policies and their operational efficiency. In other words, a higher degree of aggregation gives a lower complexity but also a lower operational efficiency of the derived policies; and a higher degree of disaggregation has the opposite effect.

Since the complexity control that is described in the previous paragraph is applied only on the policy space, the analytical solution of the resulting MP is still intractable because the evaluation of the objective function requires the underlying steady-state distribution of the system sojourn at each state. As a consequence, this MP is solved through a simulation optimization method called stochastic approximation (SA).

The aforementioned methodological framework is summarized in Figure 1.2. The depicted framework is divided into two parts: A first "modeling" part consists of the procedures to formulate the scheduling problem of a RAS into an MP, which have been briefly discussed in the previous paragraphs. The second, "algorithm" part, is

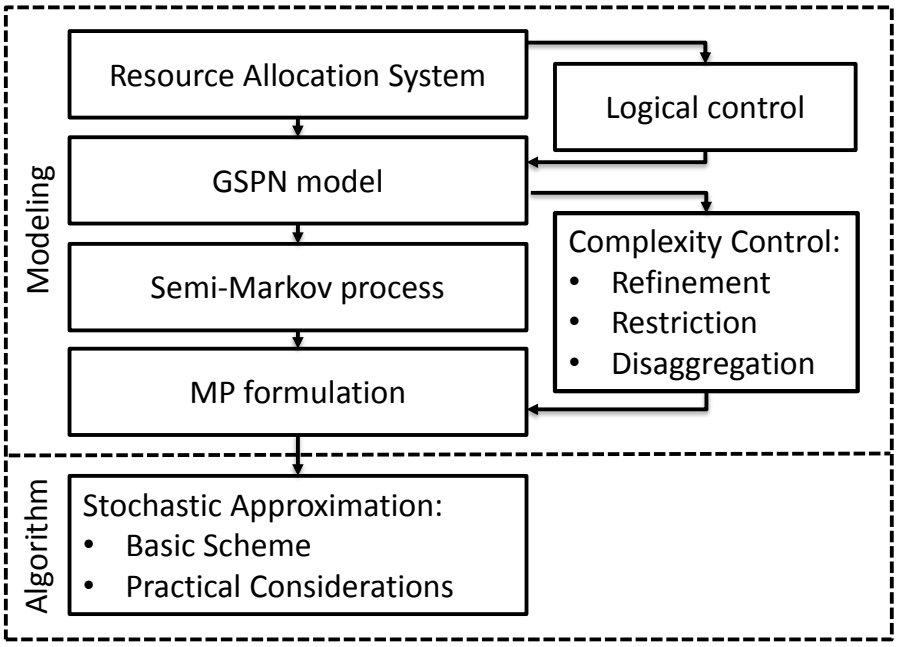


Figure 1.2: The methodological framework

focused on the SA. This part mainly seeks an adaptation to the considered problem of existing tools and results from the current literature on simulation optimization.

Although the presented methodological framework can be applied to any general RAS, this research will mainly focus on a class of RAS that abstracts the capacitated re-entrant line (CRL) model [20], and it will use this RAS class to demonstrate the overall methodology. Also, in some parts, the presented methodology will be further customized from its more general development to take advantage of the special structure of the CRL model. The CRL model itself is an abstraction of a special type of the manufacturing cell that was mentioned in the opening paragraph. In particular, this manufacturing cell supports the processing of only one type of arriving parts. So, there is one fixed order of machines, called the corresponding part "route" or "process plan". However, there is still some indeterminism coming from the fact that certain machines will support the processing of more than one stage. In other words, in the considered manufacturing cells, the route will contain some revisits, or *re-entrances*, to certain machines. Furthermore, the total number of parts at each processing

station, including those being processed, is limited by the available buffer slots at those stations. Hence, the re-entrant line is *capacitated*. Then, the problem does not involve only the allocation of the machine servers, which perform the processing of the part, but the allocation of the buffer slots as well.

From a theoretical standpoint, the throughput maximization of re-entrant lines without buffer-size limits has been well-studied: the throughput, which is decided by the feeding rate in steady state, can be maximized to (almost) the bottleneck rates among all the workstations, as long as the applied scheduling policies maintain the "stability" of the system [52]. And it has been proved that some simple policies, like the last-buffer-first-served (LBFS) and first-buffer-first-served (FBFS) policies, are stable [66, 24].

However, when the buffer sizes become finite, the introduced possibility of blocking and deadlocking effects necessitates the structural or behavior control of these systems, and simple policies such as LBFS or FBFS may not be throughput-optimal [84]. As a result, the CRL scheduling problem retains the two key objectives of a general RAS scheduling problem of logical and performance control, and it can be used as a testbed for the considered methodology. Meanwhile, the CRL is a simplified RAS since it supports only a single route, and therefore, a single process type. In the considered research program, we shall focus primarily on the problem of maximizing the CRL performance as defined by the long-term throughput of the corresponding part type. This is an *open* scheduling problem in the current literature.

On the more practical side, the re-entrant line model is widely applied as an abstraction of the fabrication operations in semiconductor manufacturing [52], and as this industry moves to higher levels of integration through the deployment of the, so called, "cluster tools" [97], the constraints on the buffer capacities that are considered in this work extend the power of the original re-entrant line model by allowing the modeling and analysis of the blocking phenomena and their consequences.

In summary, the formal models and the methods developed in this work contain the following important features:

1. the leveraging of formal DES-based representations of the RAS behavior for the systematic characterization of the target policy spaces;

2. the ability of the aforementioned representations to support a seamless integration of the RAS logical and performance-control problems;

3. the ability to integrate, in the pursued analysis, additional requirements on the behavior of the underlying system through the appropriate augmentation of the DES-based representation of the underlying RAS;

4. the ability to control explicitly the trade-off between the representational and computational complexity of the target policies and their operational efficiency; and

5. the customization of the more general developments to the CRL scheduling problem of throughput maximization, an interesting and open scheduling problem in the current literature.

Apart from the forementioned features, this work also seeks to explore deeper the practical side of the SA algorithms. Such an investigation is necessary given the complexity and the intended scale of the formulations to be addressed in this work. Hence, while acknowledging all the current results on the *asymptotic* analysis of the SA algorithms, we put more emphasis on the *practical* side, seeking to integrate to the standard SA algorithm some methods based on statistical inferences, in order to strengthen the ability of this algorithm to identify a (near-)optimal scheduling policy in a reliable, expedient and computationally cost-efficient manner.

The rest of the thesis is organized as follows: Chapter 2 formally defines the RAS model and its performance optimization problem, and also gives necessary background

information to build the methodological framework. Chapter 3 addresses the "modeling" part in Figure 1.2 except for the complexity control, which is the topic of Chapter 4. Chapter 3 also addresses the RAS-based modeling of the considered CRL operations. Chapter 4 addresses the complexity problems discovered in the process of building the MP formulation for the considered RAS scheduling problem. Chapter 5 covers the "algorithm" part in Figure 1.2. Both Chapters 4 and 5 also report results from a number of numerical experiments that demonstrate the efficacy of the presented methods. Finally, Chapter 6 concludes the thesis and discusses some remaining open issues and possible extensions.

CHAPTER II

BACKGROUND

This chapter introduces the necessary background information for the foundation of the methodological framework that was outlined in Chapter 1. First, the modeling abstraction of a sequential 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 in the context of the PN structure. The third section gives the necessary basic knowledge on the GSPN, which will be the major modeling tool for the presented methodological framework. Finally, the chapter concludes with the formal definition of the CRL scheduling problem of throughput maximization and its characterization as a RAS performance optimization problem.

2.1 Formal definition of the sequential resource allocation system

The common characteristics of the sequential resource allocation systems that are considered in this work have already been introduced in a more intuitive manner in Chapter 1. In this section, a formal description of those characteristics is cited from [86] (c.f. Section 2.1 in Chapter 1):

Definition 1 *A sequential resource allocation system (RAS) is defined by a quintuple* $\Phi = \langle \mathcal{R}, C, \mathcal{P}, \mathcal{A}, \mathcal{T} \rangle$ *where:*

1. $\mathcal{R} = \{R_1, \ldots, R_m\}$ *is the set of the system* resource types.

2. $C : \mathcal{R} \mapsto \mathbb{Z}_+$ – *the set of strictly positive integers*[1] – *is the system* capacity *function, characterizing the number of identical units from each resource type available in the system. Resources are considered 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. $\mathcal{P} = \{\Gamma_1, \ldots, \Gamma_n\}$ *denotes the set of the system* process types *supported by the considered system configuration. Each process type* Γ_j *is a composite element itself, in particular,* $\Gamma_j =< \mathcal{S}_j, \mathcal{G}_j >$*, where: (a)* $\mathcal{S}_j = \{\Delta_{j1}, \ldots, \Delta_{j,l(j)}\}$ *denotes the set of* processing stages *involved in the definition of process type* Γ_j*, and (b)* $\mathcal{G}_j$ *represents some data structure communicating some sequential logic that applies to the execution of any process instance of type* Γ_j*. The sequential logic should be defined in a way such that no re-visit to any stage happens in any paths.*[2] *For further reference, we also set* $\mathcal{S} \equiv \bigcup_{j=1}^{n} \mathcal{S}_j$*.*

4. $\mathcal{A} : \mathcal{S} \mapsto \prod_{i=1}^{m} \{0, \ldots, C_i\}$ *is the* resource requirement function[3] *associating every processing stage* Δ_{jk} *with a* resource allocation request $\mathcal{A}(j,k) \equiv A_{jk}$*. More specifically, each* A_{jk} *is an* m*-dimensional vector, with its* i*-th component indicating the number of resource units of resource type* R_i *necessary to support the execution of stage* Δ_{jk}*. It is further assumed that* $A_{jk} \neq \mathbf{0}$ *for any processing stages, where* $\mathbf{0}$ *denotes the column vectors of appropriate dimension, with all its components equal to 0.*[4] *Obviously, in a well-defined RAS,* $A_{jk}(i) \leq C_i, \ \forall j, k, i$*.*

5. $\mathcal{T} : \mathcal{S} \mapsto \mathcal{D}$ *is the* timing function*, corresponding to each processing stage* Δ_{jk}

[1] *In the sequel,* $\mathbb{Z}_+$ *will represent the set of strictly positive integers, while* $\mathbb{Z}_{0+}$ *will represent the set of non-negative integers. And the space of the superscript will be reserved for the dimension display. Similar notations will be applied to the set of real numbers* $\mathbb{R}$*.*

[2] This condition excludes the situations in which a process instance can entangle itself in an indefinite loop among its processing stages.

[3] To distinguish from the actual decision made when facing the competition among the process instances for some insufficient required resource, the expression "resource requirement function" is used instead of "resource allocation function" in the original text of [86].

[4] Similar notations such as "$\mathbf{1}$" will be applied in the sequel.

a distribution D_{jk} that characterizes the statistics of the "processing time" *t_{jk}, experienced during the execution of stage Δ_{jk}. It is possible that for some j, k, $\mathbb{P}\{t_{jk} = 0\} = 1$; for instance, the use of "buffer" resources.*

Furthermore, for purposes of complexity considerations, the size *$|\Phi|$ of RAS Φ is defined as $|\Phi| \equiv |\mathcal{R}| + |\mathcal{S}| + \sum_{i=1}^{m} C_i$.*

Next, the above definition is further elaborated by some additional assumptions that are also discussed in [23]. These assumptions are typically known as the "mutual exclusion", "wait-for", "no preemption" and "circular wait" conditions, and they constitute necessary and sufficient conditions for deadlock formation. In many applications, the first three of these conditions are determined by the system configuration itself, and the deadlock can be avoided only by avoiding the development of the fourth condition. In this document, the first three of these conditions will be expressed by the following assumptions, with the corresponding statements adapted to Definition 1. These assumptions can be seen as the "pre-conditions" where deadlock may arise. Furthermore, Assumption 1 is quite essential for the considered resource allocation function.

Assumption 1 *The resource allocation that is implied by item 4 of Definition 1 is* mutually exclusive*, i.e., if a resource unit is allocated to some stage Δ_{jk} of a process instance, then it is not available for allocation to other process instances until it is released. Or, stated differently, in the considered RAS, the resources cannot be shared.*

Assumption 2 *Once a resource is allocated to some non-terminal stage Δ_{jk} of a process instance, it cannot be released until the process instance has been allocated the necessary resource differential $(\mathcal{A}(\Delta_{jk'}) - \mathcal{A}(\Delta_{jk}))^+$ with respect to one of its successor stages $\Delta_{jk'}$, and has advanced to $\Delta_{jk'}$.*

Assumption 2 implies that (i) the process instance cannot be removed from the system temporarily to accommodate the requirements from other process instances (i.e.,

no preemption), and (ii) the release of the currently allocated resource units should *wait for* the allocation of the next required resources. This "hold-while-waiting" effect applies naturally to certain resources, such as the resources corresponding to the buffer slots in the manufacturing cell that were mentioned in Chapter 1. On the other hand, this assumption does not restrict the modeling power of the RAS. In the case that a resource unit can be released without requiring the resource allocation of any additional resource units, one can insert "fictitious" processing stages with: (i) a resource requirement vector obtained from the current resource requirement vector by removing the resource(s) to be released, and (ii) zero processing time.

As discussed in Chapter 1, the resource allocation taking place in the considered RAS must be performed in a way that ensures logical correctness and operational efficiency. In this work, the notion of logical correctness refers only to the absence of partial deadlock. The following definition characterizes the state of the system when a partial deadlock happens [23]. And it can be seen as the system state that should be avoided via the logical control policy adopted in this research program.

Definition 2 *A sequential RAS partial deadlock is defined by a RAS state containing a set of process instances that cannot advance to any of their next processing stages because each such advancement requires a set of resources that are currently held by the remaining processes in this set.*

The objective of operational efficiency for any given RAS refers to the optimization of some time-based performance criteria, through the specification of a policy that is defined on the state space of the RAS-modeling DES and will guide the selection among the set of admissible actions as in the control scheme in Figure 1.1. For the purpose of such time-based performance considerations, the following assumption, which complements item 5 of Definition 1, makes sure that no process instance can finish in zero time:

Table 2.1: A RAS taxonomy [86]

Based on the structure of the Process Sequential Logic	**Based on the structure of the Resource Requirement Function**
Linear: Each process is defined by a linear sequence of stages.	**Single-Unit:** Each stage requires a single unit from a single resource type.
Disjunctive: A number of alternative process plans encoded by an acyclic digraph.	**Single-Type:** Each stage requires a single resource type, but might engage more than one unit of it.
Coordinating: Each process is an acyclic fork-join network.	
Complex: A combination of the above behaviors.	**Conjunctive:** An arbitrary number of units from different resources.

Assumption 3 *For any process type Γ_j, in any possible realization path in the corresponding $\mathcal{G}_j$, there must exist a processing stage Δ_{jk}, such that the processing time distribution D_{jk} satisfies $\mathbb{P}\{t_{jk} = 0\} = 0$.*

The introduced RAS concept can model a rich set of structures with respect to (i) the process sequential logic and (ii) the resource requirement function that defines the resource allocation requests of the various processing stages. In particular, [86] gives the taxonomy summarized in Table 2.1.

2.2 The RAS deadlock avoidance problem

The RAS logical control problem in this research refers only to the avoidance of the partial deadlock states characterized in Definition 2. The corresponding qualitative behavior of a RAS Φ can be modeled as a finite state automaton (FSA) $G(\Phi)$ [86]. And the effective avoidance of all these deadlock states can be equivalently defined as the avoidance of the states that do not possess a path back to the "empty" state in the corresponding state transition diagram (STD). The empty state itself corresponds to the state where the RAS is empty of any activated processes, and it naturally defines the initial and the marked state of the corresponding FSA. More generally, in the literature on deadlock avoidance, the RAS state is typically defined as a vector

$s \in \mathbb{Z}_{0+}^{|\mathcal{S}|}$, where each component of s gives the number of process instances in its corresponding processing stage.[5]

A "naive" solution to the RAS deadlock avoidance problem enumerates the STD of the underlying FSA corresponding to the subspace that is reachable from the empty state, and then eliminates the events that result to *unsafe* states, i.e., to states with no paths to the empty state. As a result, the STD of the logically controlled RAS is a strongly connected digraph. Each node of this digraph is called a *safe* state.[6] Since this method eliminates only the unsafe states and keeps all the safe states, it gives a *maximally permissive* deadlock avoidance policy (DAP), which is an optimal solution for the considered logical control problem.

However, this naive solution requires enumeration of the state space and it is not practical. Furthermore, although the presence of partial deadlock in any given RAS state is polynomially detectable with respect to the RAS size $|\Phi|$ [86], the "prediction" of deadlock, or deciding whether a state is safe or not, is an NP-complete problem for most RAS classes [39, 90]. The NP-completeness of the state-safety assessment can be proved even for the simplest RAS class, characterized by linear sequential logic and a single-unit resource requirement function [58]. Hence, the computation of the maximally permissive DAP is an NP-hard problem in general, and a number of methodologies are proposed in the relevant literature to address this issue.

The above remarks further imply that for these RAS classes where the assessment of state-safety is NP-complete, the corresponding difficulty stems from the existence of "deadlock-free unsafe" states, i.e., states that do not contain any partial deadlocks

[5]This definition is proper only in the logical control problems. When timing is introduced, the dimension of s is larger than $|\mathcal{S}|$, as different process instances at the same processing stage may be in different status: some are in processing, while others are waiting for advancement to the next stage.

[6]The discussion in this document is limited to the enumeration of the state space that can be reached from the empty state. Hence, the whole state space is the set of *reachable* states. And the set of "*safe*" states in this document refers to the intersection of the sets of *safe* and *reachable* states in the terminology of most of the relevant literature.

but will definitely lead to some "partial-deadlock" states. This remark will be essential in some of the subsequent developments.

One solution to the computational challenges that are described in the previous paragraphs is the relaxation of the requirement of maximal permissiveness. In the corresponding policies [5, 54, 55, 32, 56, 57, 79, 31], some constraints on the state vectors are applied so that only safe states satisfy these constraints. But the resulting policy might be sub-optimal if the region of safe states cannot be represented by the employed constraints, and thus, some safe states are also eliminated by these constraints. In the design of such more restrictive policies, one should pay additional attention to the policy-induced, or *restricted* deadlock; a restricted deadlock is a policy-admissible (and therefore safe) state, where no further progress is possible due to the blockage of feasible actions by the policy itself [5, 32].

On the other hand, if the RAS falls into the special categories where there are no deadlock-free unsafe states, then RAS state safety becomes equivalent to absence of partial deadlock. Therefore, a one-step look-ahead method which tests whether the state that results from the execution of a feasible action is deadlocking, is able to avoid any possible unsafe states. Some examples of such RAS classes can be found in [88, 32, 58, 86]. Furthermore, the "safety region", i.e., the set of safe states, can be specified by a set of linear constraints on the RAS state in some special RAS classes. In [77], the problem of building an optimal linear classifier representing the maximally permissive DAP was modeled as a mixed integer program (MIP).

Another line of research converts the RAS deadlock avoidance problem into the problem of liveness enforcement for a RAS-modeling Petri net (PN). In fact, the problem of liveness enforcement for various PN classes is a broader problem in the DES literature that transcends the class of RAS-modeling PNs. Some of the most interesting results in this direction seek to associate the absence of liveness to the formation of certain structures in the PN state (or "marking"), and eventually prevent

these formations through the superimposition of additional structure on the original PN. Frequently, this additional structure takes the form of *monitor*, or *supervisor*, places [37, 72, 47]. In the case of RAS-modeling PNs, the underlying special structure has led to stronger structural characterization of the net liveness, and more powerful synthesis methods for the necessary supervisors. Relevant results can be traced in [29, 99, 79, 48, 85, 64, 63, 65].

Since a PN constitutes a representation of the underlying RAS that is polynomially related to the corresponding size $|\Phi|$, the aforementioned methods, when applicable, can lead to the synthesis and the deployment of the maximally permissive DAP while avoiding the (explicit) enumeration of the underlying state space. In most of these cases, the obtained policy also has a quite compact representation as a fairly small number of supervisor places. But representation of a policy by a set of supervisor places implies that it can be expressed as a set of linear constraints on the net marking [37], and this fact constitutes a substantial limitation for the aforementioned methods when it comes to the characterization and deployment of the maximally permissive DAP. So, next we turn to another approach that allows for a more general, and therefore, more complete and more effective representation of the target supervisory control policies, and constitutes the primary method of choice for the considered research program.

2.2.1 A classifier-based method for deadlock avoidance

Recently, the work presented in [74, 89] developed a method that supports the synthesis of the maximally permissive DAP for the various RAS classes considered in the work, and has promising empirical representational and computational complexity. This method contains three major phases:

The first phase is a computation of the sets of the safe and unsafe states in the

underlying STD, with methods that are similar to these employed by the aforementioned "naive" solution. However, the corresponding complexity issues are addressed in two ways:

1. This first phase is performed "offline", and the derived information will be stored for later usage in a more efficient manner. Given the offline nature of this operation, the constraints on the employed computational resources, especially the corresponding time budgets, are more relaxed.

2. In many cases, the performed enumeration is only "partial", i.e., it does not expand on the whole state space. Indeed, an effective implementation of the maximally permissive DAP can be based only on the knowledge of the "boundary" unsafe states, i.e., the unsafe states which are reachable from a safe state by one event. Furthermore, due to the monotone nature of state (un)safety in many RAS classes (i.e., the fact that if a state vector s_{u} is unsafe, then any vector s'_{u} that is componentwise no less than s_{u} is also unsafe), the entire set of boundary unsafe states can be characterized by its minimal elements. These (minimal) boundary unsafe states can be obtained through a backwards search on the underlying state space that starts with a programmatic construction of all the (minimal) deadlocks [76].

The second phase is also performed offline, and concerns the representation of the dichotomy of the RAS state space into its safe and unsafe spaces, that was computed in the first phase, by a pertinent classifier. This classifier can be either "parametric" or "non-parametric".

The *parametric* classifier is represented by a linear combination of a number of indicator functions. Each indicator function specifies whether the state vector is in a specific polyhedron. The coefficients of the linear combination and the polyhedra are determined through some computational techniques that employ the sets of maximal

safe and minimal boundary unsafe states. In the special case that the safety region can be represented by a set of linear constraints, the above classifier collapses to a set of linear inequalities and it is characterized as "*linear*". The linear combination of indicator functions may be replaced by other relationships on the underlying polyhedra, but, in any case, the basic requirement is the effective expression of the potential non-convexity of the safety region of the underlying state space. The reader is referred to [89] for more details.

The *non-parametric* classifier encodes more directly the set of all the minimal unsafe states. For instance, [75] introduced the method of (n-ary) decision diagrams, which can effectively store the entire set of the minimal unsafe states in an acyclic digraph, while avoiding significant levels of redundancy that take place in the typical storage of these vectors in an array-based representation. In fact, the representational compression that is obtained by this approach is usually at a logarithmic level. Furthermore, these data structures can also support efficient search processes for any given vectors.

Finally, the third phase in the method of [74, 89] is an online phase that enforces the maximally permissive DAP through a classical one-step look-ahead scheme. More specifically, during the operation of the considered RAS, whenever a resource allocation action is contemplated, the state resulting from the considered action will be calculated and then tested through the developed classifier. If the test determines the state as unsafe – or more generally, as *inadmissible* – then this action is blocked by the implemented DAP.

As already mentioned, in the methodological framework presented in this research program, the logical controller will generally adopt the one-step look-ahead scheme of the classifier-based method. However, we shall also encounter some special structures that may simplify the representation and the deployment of the applied DAP. In particular, if a RAS can be verified to contain no deadlock-free unsafe states, then

the "offline" phases that build the aforementioned classifier can be skipped, and the employed one-step look-ahead scheme will only test for deadlock states. Furthermore, if the safety region of a given RAS can be represented by a set of linear constraints, then a PN-based representation of the maximally permissive DAP through supervisor places is also possible. In this case, the one-step look-ahead filter on the admissible actions is no longer necessary. Instead, a control subnet, consisting of the necessary supervisor places, is added to the original RAS-modeling PN. Some examples of such results will be provided in Section 3.2, where the logical control schemes that are considered in this section are customized to the GSPN that models the CRL.

2.3 The GSPN reward model and its performance evaluation

This section gives some necessary background information on the Petri net (PN), the generalized stochastic Petri net (GSPN), and the GSPN reward models, and on the performance evaluation of the GSPN. The main focus is on the basic concepts and the corresponding notations. The reader is refered to [73] and Chapter 2 of [61] for a complete discussion on the PN properties and their underlying structural and behavioral analysis, to [1] for a complete and detailed methodology for the GSPN-based modeling, and to [1, 22] for the performance evaluation of the GSPNs.

2.3.1 Definition and basic properties of Petri nets

The Petri net (PN) is a modeling tool that is widely applied in the area of DES. The definition of a PN includes a weighted bipartite digraph $\mathcal{N}$ and a vector M_0. In particular, $\mathcal{N}$ defines the *structure* of the PN as a quadruple $\langle \mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{W} \rangle$, where: the node set is the union of two mutually exclusive subsets, the *place* set $\mathcal{P}$ and the *transition* set $\mathcal{T}$; the arc set $\mathcal{F} \subseteq \mathcal{P} \times \mathcal{T} \cup \mathcal{T} \times \mathcal{P}$ models the *flow* structure that is supported by the net; and the weight set $\mathcal{W} : \mathcal{F} \mapsto \mathbb{Z}_+$ specifies an amount of flow for each arc. On the other hand, the *behavior* of the PN is modeled by the dynamics

of the *tokens* that circulate among the places. In particular, the state of the PN is defined by the number of tokens at each place, and the vector $M \in \mathbb{Z}_{0+}^{|\mathcal{P}|}$ composed by these numbers is called the net *marking*. A special marking M_0, called the *initial marking*, defines the initial state of the PN. If the PN is used to model a DES, then the marking corresponds to the state of the DES.

In the graphical representation of a PN, the places are depicted as circles, the transitions are depicted as bars, and the flow between the places and the transitions is represented by weighted directed arcs, where the weights are explicitly stated only if they are greater than one. Tokens are depicted as black dots in the circles of the corresponding places. An example PN in graphical representation can be found in the left side of Figure 2.1 in the next section.

For a place $p \in \mathcal{P}$ and a transition $t \in \mathcal{T}$, if $(p,t) \in \mathcal{F}$, then p is an *input place* of the transition t, and t is an *output transition* of the place p; if $(t,p) \in \mathcal{F}$, then p is an *output place* of the transition t, and t is an *input transition* of the place p. The set of all input transitions of p is the *preset* $\bullet p$; the set of all output transitions is the *postset* $p\bullet$. The notations $\bullet t$ and $t\bullet$ have similar meanings. A transition t is *enabled* at a marking M if $M[p] \geq \mathcal{W}(p,t),\ \forall p \in \bullet t$. The set of all the enabled transitions at a marking M is denoted as $\mathcal{E}(M)$. Since the enabling of a transition at a given marking M is determined by the PN structure, the set $\mathcal{E}(M)$ can be seen as the set of feasible actions at M, according to the general semantics of the DES control scheme of Figure 1.1. When a transition is enabled at a marking M, then it can be fired and the marking will be changed to M', where

$$M'[p] = \begin{cases} M[p] - \mathcal{W}(p,t) & \text{if } p \in \bullet t \setminus t\bullet \\ M[p] + \mathcal{W}(t,p) & \text{if } p \in t\bullet \setminus \bullet t \\ M[p] - \mathcal{W}(p,t) + \mathcal{W}(t,p) & \text{if } p \in \bullet t \cap t\bullet \\ M[p] & \text{otherwise} \end{cases}$$

In other words, the firing of an enabled transition t first will *consume* $\mathcal{W}(p_{\mathrm{i}}, t)$ token(s) from each input place $p_{\mathrm{i}} \in {}^{\bullet}t$ and subsequently it will *release* $\mathcal{W}(t, p_{\mathrm{o}})$ token(s) to each output place $p_{\mathrm{o}} \in t^{\bullet}$. In the perspective of DES, the firing of a transition can be perceived as an event that changes the state of the system. In the following, we shall use the notation $M' = tr(M, t)$ or $M \xrightarrow{t} M'$ to denote that marking M' is the result of firing transition t at marking M. In this case, we shall also say that marking M' is *reachable* from marking M. The notions of "enabled" and "reachable" can be further extended to transition sequences in an inductive way: if $M_1 \xrightarrow{t_1} M_2$, $M_2 \xrightarrow{t_2} M_3$, ..., $M_{i-1} \xrightarrow{t_{i-1}} M_i$, then the transition firing sequence $\sigma \equiv t_1 t_2 \ldots t_{i-1}$ is *feasible* at marking M_1, and M_i is reachable from M_1, denoted as $M_i = tr(M_1, \sigma)$ or $M_1 \xrightarrow{\sigma} M_i$. The transition firing sequence can be an empty sequence, denoted by ε; therefore, a marking is always reachable from itself, i.e., $M = tr(M, \varepsilon)$, $\forall M$. For a set of transitions $\tilde{T}$, $\tilde{T}^*$ is the Kleene closure, which is the set of all the finite sequences composed by transitions in $\tilde{T}$, including the empty sequence ε. For a transition sequence $\sigma \in \tilde{T}^*$, $\vec{\sigma}$ is the Parikh vector, whose dimension is $|\tilde{T}|$ and its components count the occurrences of the corresponding transitions in σ. The set of all the markings that are reachable by a given marking M is denoted by $\mathcal{R}(M)$. Particularly, the set $\mathcal{R}(M_0)$ is also denoted as $\mathcal{R}(\mathcal{PN})$; and this set constitutes the state space of the PN. In the sequel, the dependency on $\mathcal{PN}$ will be suppressed, and the notation $\mathcal{R}$ will be adopted to denote the state space. For a PN with a given initial marking M_0, if the number of tokens of all its places can be upper-bounded for all the markings in $\mathcal{R}$, then the PN is said to be *bounded*. Obviously, the boundedness of a PN implies the finiteness of the state space, since the marking vectors are integer-valued.

The reachability relationship of all the markings of the PN can be represented by a digraph, called a *reachability graph*. In the reachability graph, a marking M is represented by a node, and the firing relationship $M \xrightarrow{t} M'$ is represented by an arc from the node representing M to the node representing M'. If the reachability graph

is finite, then it is the STD of the FSA that represents the PN dynamics.

Next, let us define the notions of "reversibility" and "liveness" of a PN:

Definition 3 *A bounded PN, $\mathcal{PN} = \langle \mathcal{N}, M_0 \rangle$, is* reversible *if its reachability graph is strongly connected, i.e., for every marking that is reachable from the initial marking M_0, there exists a feasible transition firing sequence $\sigma \in \mathcal{T}^*$ such that $M \xrightarrow{\sigma} M_0$.*

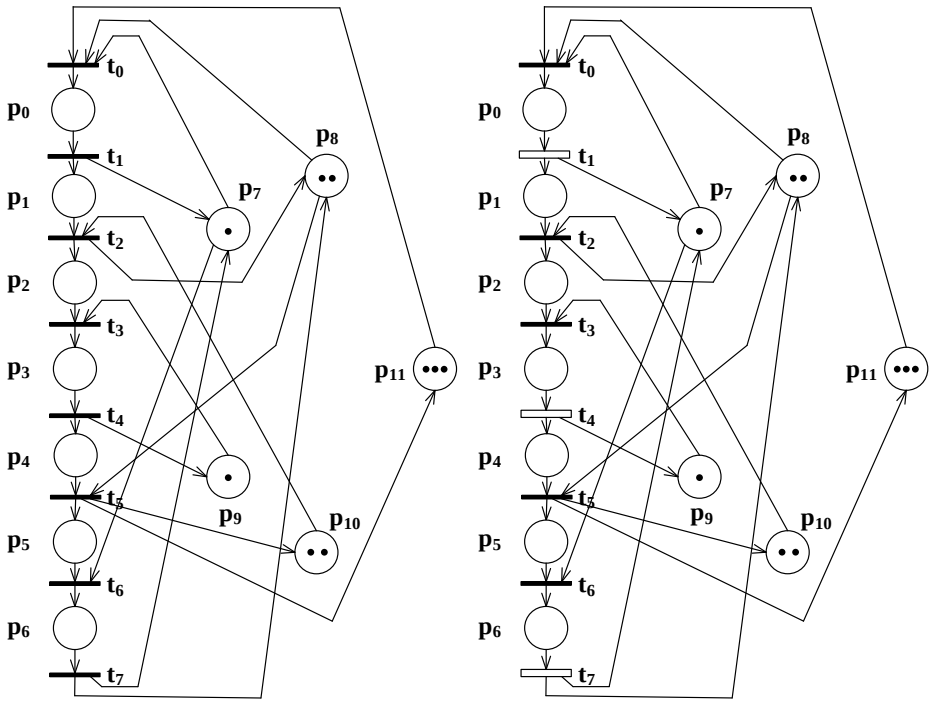
Definition 4 *A PN, $\mathcal{PN} = (\mathcal{N}, M_0)$, is* live *if for every transition $\hat{t} \in T$ and every marking $M \in \mathcal{R}$, there exists a feasible transition firing sequence $\sigma \in \mathcal{T}^*$ such that $M \xrightarrow{\sigma} M'$ and $\hat{t} \in \mathcal{E}(M')$.*

Remark For RAS-modeling PNs, both reversibility and liveness are equivalent to the absence of partial deadlock in the underlying RAS. Also, for a PN modeling the behavior of a RAS under the control of a given DAP, reversibility and liveness imply the "correctness" of that DAP, in the sense that the DAP keeps the state of that RAS in its safe region and does not induce restricted deadlock. For a more extensive discussion connecting the RAS deadlock and deadlock avoidance to the properties of reversibility and liveness of the RAS-modeling PN, the reader is referred to [85] and Chapter 5 of [86].

2.3.2 The extension of a PN to GSPN

A generalized stochastic Petri net (GSPN) extends the definition of PN by (i) partitioning the set of transitions $\mathcal{T}$ into the set of immediate or *untimed* transitions $\mathcal{T}_u$, and the set of *timed* transitions $\mathcal{T}_t$; and (ii) adding a new mapping $r : \mathcal{T}_t \mapsto \mathbb{R}_+$, defining the firing rate of the timed transitions.[7] The firing of a timed transition t will experience a delay with an exponentially distributed random time length; and the rate of this exponential distribution is $r(t)$. Also, for any marking M, the set of

[7] In the original definition of the GSPN that is provided in [1], the mapping r may have the domain $\mathcal{T} \times \mathcal{R}$, which allows dependency of the firing rates on the net markings. However, this feature is not particularly relevant to the GSPN models that are pursued in this work.



(a) An example PN (b) The PN of (a) converted to a GSPN

Figure 2.1: An example PN and its conversion to a GSPN

enabled transitions $\mathcal{E}(M)$ can be partitioned into the sets of enabled untimed transitions $\mathcal{E}_u(M)$ and enabled timed transitions $\mathcal{E}_t(M)$. In the graphical representation of a GSPN, the untimed and timed transitions are respectively depicted by black and white bars. Sometimes the timed transitions are also labeled with their firing rates.

The right part of Figure 2.1 is an example GSPN where the transitions t_1, t_4 and t_7 of the PN on the left part have been converted to timed transitions; the other transitions remain untimed. In fact, the PN and the GSPN depicted in Figure 2.1 model respectively the logical and the timed dynamics of an example RAS that will be introduced later in this chapter and will be used as a vehicle for demonstrating the various concepts and techniques that are developed in this document.

The dynamics of a GSPN can be described as follows: at a marking M, if only timed transitions are enabled, i.e., $\mathcal{E}_t(M) \neq \emptyset$ and $\mathcal{E}_u(M) = \emptyset$, then: (i) M is called a *tangible* marking; (ii) the transition t to be fired will be selected through an *exponential*

race defined by the firing rates of the enabled timed transitions; and (iii) there is a time span between the realization of marking M and the realization of the marking $tr(M,t)$, which is also defined by the random variable generated from the exponential race. On the other hand, if at least one untimed transition is enabled at marking M, i.e., $\mathcal{E}_u(M) \neq \emptyset$, then: (i) M is called a *vanishing* marking; (ii) the enabled untimed transitions have higher priorities than the timed transitions,[8] since they have zero firing time; and (iii) in case that more than one untimed transitions are enabled, an externally determined probability distribution must be provided for the selection of the untimed transition to be fired; this distribution is known as a *"random switch"* in the GSPN terminology. The set of markings reachable from a given marking M can be partitioned into the sets of tangible and vanishing markings, denoted by $\mathcal{R}_\mathcal{T}(M)$ and $\mathcal{R}_\mathcal{V}(M)$, respectively. For the special case of $M = M_0$, $\mathcal{R}_\mathcal{T}$ and $\mathcal{R}_\mathcal{V}$ denote the respective sets of all tangible and all vanishing markings that partition the state space $\mathcal{R}$ of the GSPN.

The timed transitions of the GSPN model enable the modeling of DES with Markovian behavior. For instance, if the processing time of each workstation in a manufacturing cell is exponentially distributed, then each processing stage can be modeled as a timed transition. Even more interestingly, some more general distributions, such as the Erlang distribution, the hypoexponential and the hyperexponential distributions, and the more general phase-type distribution, can also be modeled by an appropriately structured GSPN with its timed transitions corresponding to the exponential stages of these distributions [17]. Also, more general distributions with positive support can be approximated to any desired accuracy by a phase-type distribution [3]. Therefore, a GSPN structure can approximate these more general distributions at the cost of some additional complexity for the structure of the resulting network.

[8]It should be clarified that even if the untimed transitions have the higher priorities at a vanishing marking M, it is possible that some timed transitions are also enabled; i.e., $\mathcal{E}_t(M) \neq \emptyset$. But the timed transitions are never fired at the vanishing markings.

A random switch corresponding to a vanishing marking M contains two pieces of information: (i) the *enabling pattern* $\mathcal{E} \equiv \mathcal{E}_u(M) \subseteq \mathcal{T}_u$, which is the set of enabled untimed transitions, and (ii) the associated *selection probability distribution* that is specified by a vector $Z_M \in \mathbb{R}_{0+}^{|\mathcal{T}_u|}$, i.e.,

$$\mathbb{P}\{t \text{ is fired} \mid \text{current marking is } M\} = Z_M[t]$$

Note that in the vector Z_M, the components $Z_M[t]$ where $t \in \mathcal{E}_u(M)$ can be either zero or positive, while the other components are all zero.

2.3.3 GSPN reward models and their performance evaluation

The performance measures of GSPNs vary according to their applications. In the original literature that introduced the GSPN model [2], the reward model is not explicitly defined. Nevertheless, the typical GSPN performance evaluation often refers to the evaluation of the long-run, or "steady-state", performance measures.

In [22] more specific examples of steady-state performance measures are given, such as the frequency of firing of a particular transition, the probability that a set of places are all empty, the probability that a set of transitions are all enabled, etc. In general, two types of rewards can be adopted as the basis for the steady-state performance measures. One type associates the reward rates to tangible markings, and this type of reward model can be applied in the case where the reward rate is related to the numbers of tokens at some places or the enabling status of a set of transitions. The other type associates the rewards to transitions, and the association can be defined as either an immediate reward when a transition is fired, or a reward rate[9] associated with a timed transition. In this work, the immediate rewards associated with untimed transitions are not considered. Then, the remaining types of reward models can be unified: both the reward rates and one-time rewards associated to timed transitions

[9]The reader should notice that this can be a different rate than the firing rate.

can be converted in the form of reward rates at tangible markings, in addition to original reward rates defined on these markings (e.g., the reward rates associated to the numbers of tokens). Therefore, the GSPN reward model to be employed in this work eventually can be represented as the GSPN model described in Section 2.3.2, plus a vector $\hat{r} \in \mathbb{R}^{|\mathcal{R}_\mathcal{T}|}$ whose components are the reward rates of the tangible markings. We note that the reward rates can be negative, which should be interpreted as cost rates. Finally, the steady-state average reward η of the GSPN is the inner product between the vectors defining the reward rates $\hat{r}$ and the steady-state distribution π of the tangible markings, i.e., $\eta \equiv \sum_{M \in \mathcal{R}_\mathcal{T}} \hat{r}[M]\pi[M]$.

To obtain the steady-state distribution of the tangible markings, the first step is modeling the timed behavior of a GSPN as a stochastic process. More specifically, if all the components of a GSPN, including the bipartite digraph net structure $\mathcal{N}$, the initial marking M_0, the sets $\mathcal{T}_u$ and $\mathcal{T}_t$ of the timed and untimed transitions, the firing rates r and the pricing (or some mechanisms that can determine the pricing) of all the random switches, are given, then its behavior can be modeled as a semi-Markov process (SMP). In such an SMP, there is a one-to-one relationship between the state spaces of the GSPN and the SMP. The sojourn times of the SMP states corresponding to vanishing markings are zero, while the sojourn times of the tangible markings are exponentially distributed random variables defined by the exponential race among the enabled timed transitions at this marking. And the embedded Markov chain (EMC) of the SMP is defined by the exponential races running on the tangible markings and the random switches defined on the vanishing markings. Furthermore, it is obvious that the following two conditions are sufficient for the steady-state distribution of the corresponding SMP to be well-defined:

Condition 1 *The GSPN is bounded, or equivalently, the state space is finite, i.e.,* $|\mathcal{R}| = |\mathcal{R}_\mathcal{T}| + |\mathcal{R}_\mathcal{V}| < \infty$.

Condition 2 *The EMC of the SMP is ergodic.*

[1] calculates the steady-state distribution of the EMC of the SMP first, and then obtains the steady-state distribution of the SMP as the proportion of the average sojourn time that is spent in each state by the SMP during a regenerative cycle.

It is also possible to reduce a given GSPN to a stochastic Petri net (SPN) which consists of only timed transitions with exponentially distributed delays for their firing times [18].[10] The timed dynamics of this SPN define naturally a continuous-time Markov chain (CTMC), denoted by $\mathcal{M}$. The state space of $\mathcal{M}$ consists of the tangible markings of the original GSPN. Furthermore, for any two markings M and M' in the state space of $\mathcal{M}$, if there exists a timed transition t such that $M \xrightarrow{t} M'$, then the transition rate from M to M' is the rate of t; otherwise, the transition rate from M to M' is zero. With the infinitesimal generator Q of the CTMC $\mathcal{M}$ readily obtained, the procedure to evaluate the steady-state distribution π is standard and straightforward [93]:

$$\pi^T Q = \mathbf{0}^T$$

$$\pi^T \mathbf{1} = 1$$

In fact, the step of SPN modeling can be bypassed, and a CTMC $\mathcal{M}$ can be defined on the tangible markings of the GSPN while the vanishing markings are eliminated from the state transition diagram (STD) of the SMP built from the GSPN. Some algorithms of building the CTMC $\mathcal{M}$ from a GSPN are summarized in [22]. In this research program, and in an effort to cope with the potentially explosive size of the underlying state space, we focus on "online" algorithms that enable a sample-path based transition rate evaluation and do not require the complete enumeration of the state space. Such algorithms eliminate the vanishing markings between two tangible markings and compute the transition rates for the corresponding "macro-transition"

[10] To reduce a GSPN with fixed random switches into an SPN, some additional requirements on the structure of the GSPN are needed, since GSPNs can model a broader range of DES than SPNs. These requirements can be found in [18].

through a partial reconstruction of the underlying STD. Therefore, the different visits to tangible markings are de-coupled, i.e., the computation of the transition rates from a visited tangible marking to other tangible markings is independent from any previous similar computation. These algorithms limit the computer memory usage when the size of the state space is intractable.[11] In our work, the development of such algorithms is further facilitated by the following condition:

Condition 3 *In the considered GSPN, for any vanishing marking $M \in \mathcal{R}_{\mathcal{V}}$, there does not exist a feasible non-empty untimed transition firing sequence $\sigma \in \mathcal{T}_u^*$ such that $M = tr(M, \sigma)$.*

Under Condition 3, the transition rate from a tangible marking $M_{\mathcal{T}}$ to other tangible markings, which is a row of the corresponding infinitesimal generator matrix Q, can be determined in three steps:

1. For each enabled timed transition $t \in \mathcal{E}_t(M_{\mathcal{T}})$, there is a rate $r(t)$ and the resultant marking $\hat{M} \equiv tr(M_{\mathcal{T}}, t)$, where $\hat{M}$ can be either tangible or vanishing.

2. For each marking $\hat{M}$ obtained from the previous step, we enumerate every feasible untimed transition firing sequence $\sigma \in \mathcal{T}_u^*$ such that $M'_{\mathcal{T}} \equiv tr(\hat{M}, \sigma) \in \mathcal{R}_{\mathcal{T}}$. We also compute the products $x_{\hat{M},\sigma}$ of the selection probabilities for the transitions that appear in the aforementioned sequences σ; these probabilities are defined by the random switches at the corresponding markings that are visited by each transition sequence σ.

3. Finally, we calculate the transition rate

$$Q[M_{\mathcal{T}}, M'_{\mathcal{T}}] = \sum_{t \in \mathcal{E}_t(M_{\mathcal{T}})} r(t) \sum_{\sigma : \hat{M} = tr(M_{\mathcal{T}}, t) \wedge M'_{\mathcal{T}} = tr(\hat{M}, \sigma)} x_{\hat{M},\sigma} \tag{1}$$

[11] However, the storage space saving is not a "free lunch": the visited vanishing markings are eliminated immediately and this transition information cannot be used in the future when these vanishing markings are visited again. This effect will impact negatively the computational efficiency. There is a trade-off between storage and execution time [22].

Algorithm 1 Recursively explore the feasible untimed transition firing sequences from a marking M to tangible markings

Input: The GSPN structure and random switches, current marking M.
Output: The set of firing sequence and probability pairings $pair = \{(\sigma, x)\}$.

```
1: while |E_u(M)| = 1 do
2:    t ← the only untimed transition in E_u(M).
3:    M ← tr(M, t).
4: end while
5: if M ∈ R_T then
6:    return {(ε, 1)}.
7: else
8:    pair ← ∅.
9:    for each t ∈ E_u(M) do
10:      pair' ← recursively call this procedure with M substituted by tr(M, t).
11:      for each (σ, x) ∈ pair' do
12:         pair ← pair ∪ {(tσ, Z_M[t] · x)}.
13:      end for
14:   end for
15:   return pair.
16: end if
```

The second step can be implemented through a recursive algorithm, which is listed in Algorithm 1.

2.4 The capacitated re-entrant line, its RAS-based representation, and the CRL scheduling problem of throughput maximization

As already mentioned in Chapter 1, the capacitated re-entrant line (CRL) is a workflow model that modifies the original re-entrant line model of [52]. A CRL is a system with m workstations, indexed with $1, \ldots, m$. Workstation $i, i = 1, \ldots, m$, has a single server and B_i buffer slots. The system supports one process type with $n > m$ stages. The process type can be expressed by a mapping $\mathcal{WS} : \{1, \ldots, n\} \mapsto \{1, \ldots, m\}$, where $\mathcal{WS}(j)$ returns the index number of the workstation that supports the processing of the j-th stage. A job instance loaded to the system should complete all the processing stages sequentially to be unloaded as a final product from the system. At each processing stage J_j, the job instance should be processed at the server for

a random time span with a distribution $\mathcal{D}_j$ with positive support. Since the CRL model is mainly used for demonstration of the proposed methodology, $\mathcal{D}_j$ is supposed to be an exponential distribution with rate μ_j, for simplicity.

Assumption 4 *For the CRL class considered in this document, the processing times of the servers are exponentially distributed. And the rate for the processing stage J_j is denoted as μ_j.*

The job instance is processed in situ, i.e., when it is being processed at a workstation, it is also occupying one buffer slot. On the other hand, the transfer times among the different workstations are assumed zero, since usually they are negligible with respect to the aforementioned server processing times.

From the perspective of the sequential RAS model discussed in the previous sections of this chapter, a CRL can be modeled as a RAS $\Phi(CRL)$. This RAS contains $2m$ resource types, which are the servers and the buffer slots at the m workstations. And the capacity of each server is 1, while the capacity of the buffer slots for workstation i, $i = 1, \ldots, m$, is B_i. To define the resource requirement function regarding the servers, each original stage J_j of the CRL is further divided into three RAS stages, $\Delta_{j\mathrm{w}}$,[12] $\Delta_{j\mathrm{p}}$ and $\Delta_{j\mathrm{b}}$, which respectively capture the three phases of "waiting for processing", "being processed" and "blocked and waiting for transfer", that a job instance may experience at each CRL stage. Among the three types of the newly defined RAS stages, only the stages of "being processed" type require the allocation of the servers, i.e., the stage $\Delta_{j\mathrm{p}}$ requires the server resource of the workstation $\mathcal{WS}(j)$ in addition of one buffer slot; while the stages $\Delta_{j\mathrm{w}}$ and $\Delta_{j\mathrm{b}}$ require only one buffer slot of workstation $\mathcal{WS}(j)$. Due to the assumption of zero transfer time, the last RAS stage $\Delta_{n\mathrm{b}}$ can be merged into the stage $\Delta_{n\mathrm{p}}$, which reflects the fact that the job

[12] In the subscript "jw", "j" is a *variable* that represents the index number of the stage, while "w" is a *label-type value* representing the status of "waiting". In the sequel, the letters for the label-type values are non-italic to distinguish from the letters representing the variables.

instances that have completed their last CRL processing stages can be unloaded from the system immediately. For each CRL processing stage J_j, the timing distributions of the corresponding RAS stage representing "waiting" or "blocked" phases are the constant zero, while the timing of the stage $\Delta_{j\mathrm{p}}$ is exponentially distributed with rate μ_j. The RAS $\Phi(CRL)$ is *linear* and *conjunctive* in the RAS taxonomy depicted in Table 2.1.

The objective of the CRL scheduling problem is the maximization of the long-run throughput, while avoiding the formation of the deadlock.

The definition of the maximum long-run throughput is not exactly the same as defined in the context of the classical re-entrant line literature. For instance, [52] defines the maximum throughput as the upper bound of the feeding rate of the raw materials such that the manufacturing system remains stable, or there is no accumulation of jobs in the system in the long term. However, the finite buffer sizes of the CRL imply that the total number of jobs in the system has already been upper bounded and will not accumulate in the long term. Therefore, this work adopts the notion of the "maximum long-run throughput "defined in [20]: this is the maximum possible steady-state production rate of the CRL when infinite jobs are waiting outside of the system to be loaded. Since there are always infinite jobs waiting before the first CRL stage, it is not necessary to use the buffer slots of the workstation $\mathcal{WS}(1)$ to accommodate the job instances waiting to be processed in the first CRL stage. Therefore, the RAS stage Δ_{1w} can be merged into the RAS stage Δ_{1p}. Finally, the RAS $\Phi(CRL)$ contains a single process type with $3n-2$ processing stages, where n is the number of CRL stages.

On the other hand, while solving for the deadlock avoidance policy (DAP) in the CRL context, a simpler RAS model than $\Phi(CRL)$ can be abstracted. More specifically, the release of the server resources in $\Phi(CRL)$ takes place upon the transition from a stage $\Delta_{j\mathrm{p}}$ to $\Delta_{j\mathrm{b}}$, and it should be noticed that no new resources are required

to be allocated during this transition. Therefore, the "hold-while-waiting" condition will never happen in the allocation of the server resources, and therefore, deadlock cannot happen due to the server allocation. As a result, the server allocation can be omitted in the RAS model that considers only the logical control. The RAS model that considers only the buffer allocation is denoted as $\hat{\Phi}(CRL)$, and it contains m resource types corresponding to the buffer slots of the m workstations. Furthermore, the only process type is a linear process with n processing stages, corresponding to the n stages of the CRL; the advancement to processing stage j requires one unit of buffer slot at workstation $\mathcal{WS}(j)$ and will release one unit of buffer slot at workstation $\mathcal{WS}(j-1)$ if j is not the first stage. Modeling of timing is not necessary since only logical control is considered in the RAS model $\hat{\Phi}(CRL)$. The RAS $\hat{\Phi}(CRL)$ will be referred to as the *simplified RAS* in the sequel.

Note that there is a function defined between the state spaces of the RAS models $\Phi(CRL)$ and $\hat{\Phi}(CRL)$ that are constructed from the same CRL. If the state s of $\Phi(CRL)$ is represented by a $(3n-2)$ dimensional vector whose components are the numbers of job instances at each RAS stage, then for each such state, a corresponding state $\hat{s}$ of $\hat{\Phi}(CRL)$ can be found with the same allocation of job instances among the CRL stages. But it is also possible that different $\Phi(CRL)$ states map to the same $\hat{\Phi}(CRL)$ state. More importantly, any DAP generated from the state space of $\hat{\Phi}(CRL)$ is applicable to the state space of $\Phi(CRL)$, through the aforementioned mapping. Determining a DAP for $\hat{\Phi}(CRL)$ is much simpler than performing the same task in the context of $\Phi(CRL)$, since the resource requirement function of $\hat{\Phi}(CRL)$ is single-unit, according to the RAS taxonomy introduced at the end of Section 2.1.

The next example intends to illustrate the above definition regarding the CRL concept and its RAS-based modeling. The same example will also be used to illustrate the entire methodology that is developed in the rest of this document.

Example 1 *[59, 60] A small flexibly automated production cell.*

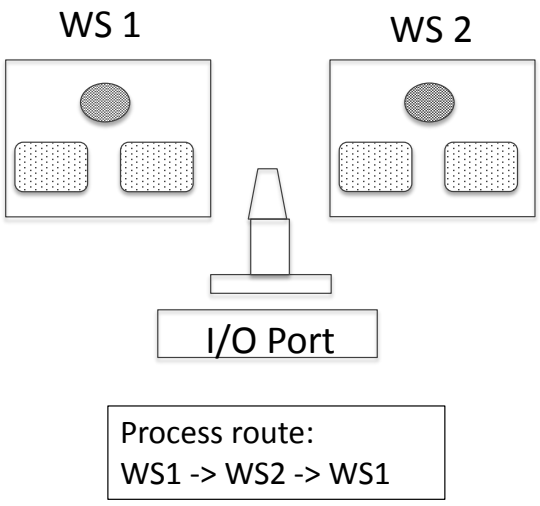


Figure 2.2: The CRL model of Example 1

We consider a flexibly automated production cell with two different machines that can process parts, and a robot that can transfer the parts among the machines and the I/O port of the cell in negligible time. Each machine has a single server and two buffer slots, and the jobs are processed in situ. This production cell supports only one process plan with three sequential steps. Once a part is loaded to the cell, it should be processed by machine 1; then the robot will move the part to machine 2 for the second step of processing; after the part finishes processing at machine 2, it will be moved back to machine 1, and the third processing step performed by machine 1 is applied on the part; finally, the part that completes all the three steps can be unloaded from the cell. The processing times for the three processing steps are exponentially distributed with rates μ_1, μ_2 and μ_3, respectively.

The production cell can be modeled as a CRL with $m = 2$ workstations and $n = 3$ stages, where $B_1 = B_2 = 2$, $\mathcal{WS}(1) = \mathcal{WS}(3) = 1$, $\mathcal{WS}(2) = 2$, and D_j is exponentially distributed with rates μ_j, $j = 1, 2, 3$. For the objective of maximization of the cell throughput, it is assumed that an infinite number of outstanding parts are waiting outside the system. The CRL model is depicted in Figure 2.2.

Furthermore, the RAS model $\Phi(CRL)$ contains $2m = 4$ resource types. Two of them are server resources with capacities 1, and the two others are buffer slot resources with capacities 2. There is one process type in $\Phi(CRL)$ containing $3n - 2 = 7$ stages, including 3 stages with $\Delta_{j\mathrm{p}}, j = 1, 2, 3$, that correspond to the phase "being processed" and have exponentially distributed processing time; 2 zero-processing-time stages $\Delta_{j\mathrm{w}}, j = 2, 3$, that correspond to the phase "waiting for processing" and 2 other zero-processing-time stages , $\Delta_{j\mathrm{b}}, j = 1, 2$, that correspond to the phase "blocked and waiting for transfer". For a given processing stage J_j with $j = 1, 2, 3$, the RAS stages $\Delta_{jx}, x =$ w, p, b, require one unit of buffer slot resource of workstation $\mathcal{WS}(j)$, while the RAS stage $\Delta_{j\mathrm{p}}$ requires, in addition, one unit of the other set of the RAS resources, namely, the server of workstation $\mathcal{WS}(j)$. The state of the $\Phi(CRL)$ can be represented by a vector $s \in \mathbb{Z}_{0+}^7$, where each component of the vector represents the number of job instances at the corresponding stage.

The simplified RAS model $\hat{\Phi}(CRL)$, which will be employed for logical control only, contains $m = 2$ buffer slot resource types. The process type contains 3 stages. Stages 1 and 3 require one unit of the buffer slots of workstation 1, while Stage 2 requires one unit of the buffer slots of workstation 2. The state of $\hat{\Phi}(CRL)$ can be represented by a vector $\hat{s} \in \mathbb{Z}_{0+}^3$. The mapping from a state s of the complete RAS model $\Phi(CRL)$ to the state $\hat{s}$ of the simplified RAS model $\hat{\Phi}(CRL)$ is as follows:

$$\hat{s}[j] = \begin{cases} s[1\mathrm{p}] + s[1\mathrm{b}] & \text{if } j = 1 \\ s[2\mathrm{w}] + s[2\mathrm{p}] + s[2\mathrm{b}] & \text{if } j = 2 \\ s[3\mathrm{w}] + s[3\mathrm{p}] & \text{if } j = 3 \end{cases}$$

In the above development of Example 1, a CRL is abstracted from an industrial application of a flexibly automated production cell. And from the CRL model, a complete and a simplified RAS models are constructed. A mapping between the state spaces of the two RAS models is also established. Example 1 will be revisited

and expanded several times in the sequel, as the methodology for RAS performance optimization is being developed.

CHAPTER III

FORMAL MODELING OF THE TIMED RAS DYNAMICS AS A GSPN, AND THE CORRESPONDING PERFORMANCE CONTROL PROBLEM

In this chapter, first we model the timed dynamics of the considered RAS as a GSPN, and we integrate to the developed GSPN model the necessary logical and performance control policies. Subsequently, this modeling method is customized in the context of the CRL scheduling problem of throughput maximization. In the third part of the chapter, we provide a formal description of the performance optimization problem addressed in this work as a mathematical programming problem, and we also discuss the representational and computational complexity of this formulation. Finally, the chapter concludes with some more technical remarks regarding a potential sub-optimality that can be incurred by the solution space of the aforementioned formulation, and the presentation of a method that can address this issue.

3.1 GSPN-based modeling of the timed behavior of the RAS

As discussed in Chapter 2, the qualitative behavior and the necessary logical control of a RAS Φ of Definition 1 can be modeled as a Petri net (PN). Some early efforts in this direction have led to the definition of the PN classes of Systems of Simple Sequential Processes with Resources (S^3PR) [29] and linear S^3PR ($L - S^3PR$) [30], while a more recent and more general model is that of S^*PR [31]. And the considered GSPN models inherit the ability of the basic PN model to express additional requirements for the behavior / operational logic of the underlying system. One can implement these additional requirements with appropriate augmentation of the net structure.

More generally, the introduction of the GSPN model to be developed in this chapter formalizes the RAS scheduling problem with a more extensively studied model (i.e., PN models) and a more complete set of structural and behavioral semantics than the original RAS abstraction defined in Section 2.1.

However, although the methodological framework based on the GSPN model establishes a foundation and enables the potential to perform the integration of the applied logical control, in the work presented in this document, the logical control is not derived from a PN-based representation of the underlying RAS and this policy itself will not necessarily be represented in the PN modeling framework. On the other hand, we shall use the GSPN structure in order to capture and analyze the impact of the underlying RAS structure on the performance-oriented control. So, the presumed constraints for the structure and the behavior of the considered RAS are only the ones assumed in Definition 1 and the three assumptions in Section 2.1. Furthermore, the basic method that will model the aforementioned RAS to a GSPN is similar to the standard methods that have been employed in the past literature for the same purpose [86]. Next, we provide a detailed account of this modeling method, and highlight it with some examples.

PN modeling of RAS This modeling can be performed in three steps:

1. Each process type Γ_j is modeled as an independent network. In particular, each stage Δ_{jk} can be modeled as a place; each disjunctive immediate precedence relationship can be modeled as a transition; and the places and the transitions are connected according to these precedence relationships. Furthermore, a "split" (resp., "merge") relationship among a set of processing stages can be modeled by a single transition with multiple output (resp., input) places. All the places

created in this step are called *process places* in the sequel.[1] Finally, the initial marking for those process-type modeling networks contains no tokens, to represent the empty state of the RAS.

2. Each resource type is modeled as a place, which is called a *resource place* in the sequel. The initial marking for these places assigns a number of tokens equal to the corresponding resource capacities.

3. The resource requirement function can be modeled by connecting the resource places with the corresponding transitions that model the resource allocation and release, in the process-type networks. It should be noted that the resource allocations and releases are not exactly the same with those implied by the resource requirement function, but they model the differential allocation and deallocation, as defined by the resource requirements of consecutive processing stages, due to Assumption 2. More specifically, for a transition t that models the advancement from stage Δ_{jk} to stage $\Delta_{jk'}$ of the same process type Γ_j, the input places in $\bullet t$ include the process place for Δ_{jk} and all the resource places corresponding to the positive components of $(\mathcal{A}(\Delta_{jk'}) - \mathcal{A}(\Delta_{jk}))$; furthermore, the weights for the corresponding arcs are exactly the value of these components. On the other hand, the output places in $t\bullet$ include the process place for $\Delta_{jk'}$ and the places corresponding to the positive components in $(\mathcal{A}(\Delta_{jk}) - \mathcal{A}(\Delta_{jk'}))$. Finally, there are no connections between t and the resource places corresponding to the zero components in $(\mathcal{A}(\Delta_{jk}) - \mathcal{A}(\Delta_{jk'}))$.[2]

[1] In some literature, these places are also called "operation" places [6, 62].

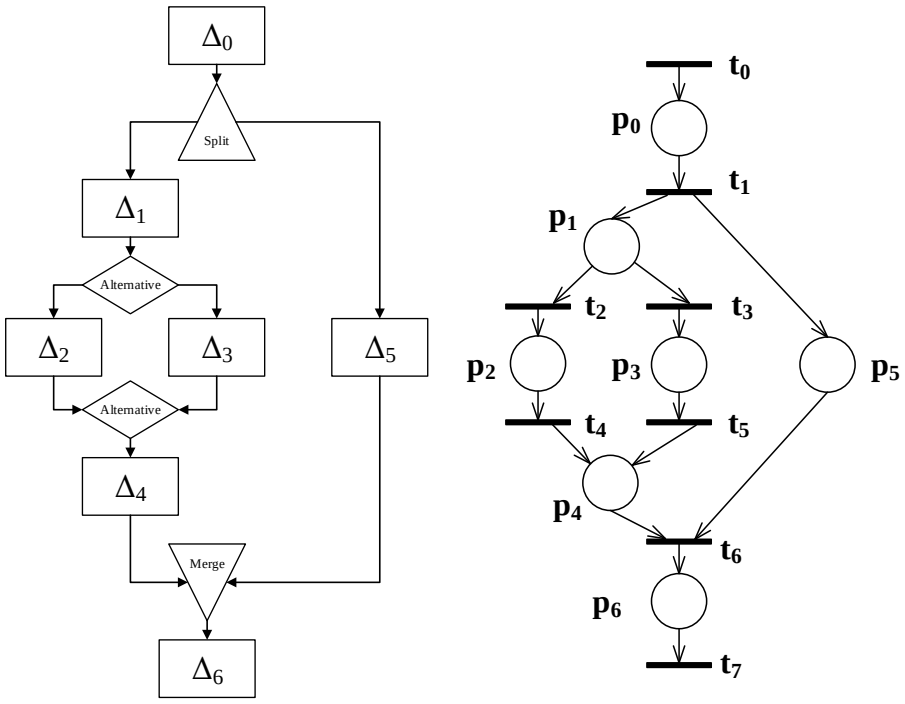
[2] This may happen even if the corresponding components of the resource requirement vector is non-zero. For instance, if the consecutive stages Δ_{jk} and $\Delta_{jk'}$ of the same process type Γ_j require the same number of units of the same type of resource R_i, then according to Assumption 2 the advancement from Δ_{jk} to $\Delta_{jk'}$ does not require extra units of R_i. As a result, there should be no connection between the resource place for R_i and the transition that models this advancement.

A more concrete illustration of the above PN structure can be found at the end of Section 3.2, where the CRL of Example 1 is modeled as a GSPN. However, since a CRL is essentially a linear RAS, Example 1 cannot illustrate the PN structure modeling RAS cases with more complex sequential logic. Therefore, next we provide Example 2. The main purpose of this example is to illustrate the PN modeling of complex sequential logic in the considered RAS.

Example 2 *A RAS process type with complex sequential logic.*

Consider a RAS process type with the sequential logic depicted in the left side of Figure 3.1. More specifically, a "split" follows the first processing stage Δ_0, where a process instance that completes the stage Δ_0 will split into two process instances. One process instance goes into the stage Δ_1; the other instance goes into the stage Δ_5. For a process instance that completes stage Δ_1, one of the stages Δ_2 or Δ_3 can be the next stage. Process instances that complete the stages Δ_2 or Δ_3 will proceed to the stage Δ_4. Finally, a process instance that completes stage Δ_4 must merge with another process instance that has completed stage Δ_5 to one process instance. This new process instance must proceed to stage Δ_6 and eventually exit the system, terminating the whole process.

In the PN model depicted in the right side of Figure 3.1, the "split" relationship is modeled by the presence of multiple output process places p_1 and p_5 for transition t_1. The "disjunctive" relationships are represented by the multiple output transitions t_2 and t_3 for the process place p_1 and the multiple input transitions t_4 and t_5 for the process place p_4. The "merge" relationship is represented by the multiple input process places p_4 and p_5 for transition t_6. With respect to modeling the relevant resource requirement for each transition $t_0, \ldots, t_7$, we further notice that since the merge/split relationships are modeled by a transition with multiple input/output process places, as in the case of transitions t_6 or t_1, the resource requirement differential that was



(a) The sequential logic of a RAS process type

(b) The PN model for the sequential logic of the depicted RAS process type

Figure 3.1: The sequential logic for the RAS process type considered in Example 2, and its PN model

discussed in the previous paragraph must be determined with respect to the total resource requirement over these places. On the other hand, since the disjunctive relationship is modeled by one transition for each option, as in the case of transition t_2, t_3, t_4 or t_5, the corresponding resource requirement differential can be modeled independently on each transition for each option.

The introduction of timed dynamics to the RAS-modeling PNs Given a processing stage Δ_{jk}, if the timing distribution D_{jk} is the constant zero, then the modeling of this stage by a single process place, as in the case of the untimed PN model, is sufficient also for modeling the timed dynamics of this stage. Otherwise, an additional transition or a subnet structure is needed to model the actual "processing" of the stage. Note that if the considered processing time is exponentially distributed, then it can be modeled by a single timed transition in the GSPN framework. Other

distributions can be approximated by some phase-type distributions to any desired degree of accuracy, and the latter can be modeled by a subnet structure that will replace the timed transition in the developed GSPN model.

Next, let us suppose that the positive processing time D_{jk} of the stage Δ_{jk} is exponentially distributed, and this processing activity is modeled by a timed transition $t_{jk\mathrm{p}}$ with rate equal to the rate of D_{jk}. Then, due to the assumption of the "hold-while-waiting" effects, the place that models the processing stage Δ_{jk}, denoted as p_{jk}, can be split into two process places, $p_{jk\mathrm{p}}$ and $p_{jk\mathrm{b}}$, which are respectively the input and output places of transition $t_{jk\mathrm{p}}$. The place $p_{jk\mathrm{p}}$ models the status under which the process instance has been allocated the necessary resources and begins processing; all the input transitions of p_{jk} are inherited by $p_{jk\mathrm{p}}$. Meanwhile, the place $p_{jk\mathrm{b}}$ models the status where the process instance has completed its processing but it still occupies the resources of the stage Δ_{jk} since it is blocked and waiting for the allocation of the resources to advance to its next processing stage(s); all the output transitions of p_{jk} are inherited by $p_{jk\mathrm{b}}$. Example 2 illustrates this modeling procedure in a more concrete manner.

Example 2 revisited Suppose that stage Δ_1 in the RAS of Example 2 has an exponentially distributed processing time with rate r_1. Thus, a timed transition $t_{1\mathrm{p}}$ is added to the network structure. Furthermore, the place p_1 is split into $p_{1\mathrm{p}}$ and $p_{1\mathrm{b}}$, which are respectively the input and output places of $t_{1\mathrm{p}}$. Then, the input transition t_1 of p_1 becomes the input transition of $p_{1\mathrm{p}}$, and the output transitions t_2 and t_3 of p_1 become the output transitions of $p_{1\mathrm{b}}$. Meanwhile, all the arcs that represent the resource allocation or release, or the sequential logic with respect to the precedent or subsequent stages, are kept on the corresponding transitions. This procedure that introduces the timed dynamics is depicted in Figure 3.2. In this figure, resource allocations and releases are omitted in the left subfigure for the original PN.

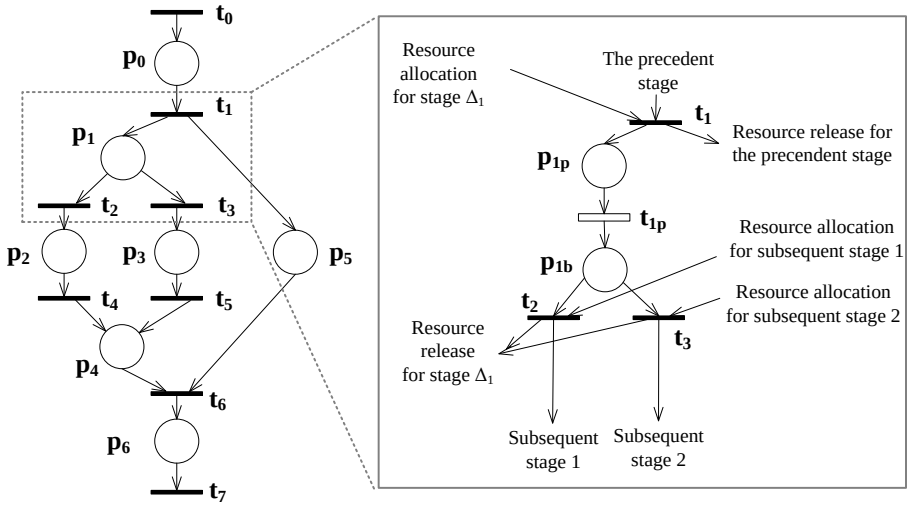


Figure 3.2: Introduction of the timed behavior in Example 2

Remark A reachable marking of a RAS-modeling GSPN is uniquely determined by the numbers of tokens in the process places, while the numbers of tokens in the resource places are determined by the resource capacity function C, the numbers of tokens in the process places, and the resource requirement function $\mathcal{A}$.[3] Because of this feature, in the sequel, we shall define a GSPN marking by providing only its submarking that corresponds to the process places. □

Next, we turn to some important aspects of the RAS-modeling GSPNs in preparation for the further developments of this document. These aspects include the integration of the RAS logical control, the specification of the performance control/scheduling policies and policy spaces, and some features that can simplify the further analysis of the underlying stochastic process.

3.1.1 Control elements for the RAS-modeling GSPN

Integrating the RAS logical control to the GSPN model The major RAS logical control methods have already been discussed in Section 2.2. To apply these

[3]In more technical terms, this possibility is due to the fact that the connectivity of each resource place to the transitions of the RAS-modeling GSPN defines an *invariant* for the net marking that involves this resource place and the process places that utilize the resource [86].

logical control methods on a RAS-modeling GSPN, we consider two forms of the representation for the RAS logical controller. The first one is applicable to the FSA-based RAS-modeling framework, but it can be easily adapted to the PN-based RAS-modeling context due to the one-to-one relationship between a PN marking and the FSA state of any given RAS. And this method is the one that is adopted in this work, in general, since it applies to the RAS whose safe and unsafe regions might not be linearly separable. The second form is only applicable when the applied control policy can be expressed as a set of linear constraints on the marking of the RAS-modeling PN, but it is valuable due to the simplicity of the resultant behavioral analysis.

1. **One-step look-ahead test**: Each enabled untimed transition of the current marking M is tested, whenever $|\mathcal{E}_u(M)| \geq 1$.[4] The test is through a simulated firing of the transition, and the examination of the policy admissibility of the resultant marking. In some special RAS configurations with no deadlock-free unsafe states, the above test can be a deadlock-detection test, which has a polynomial complexity [86]. In more general RAS configurations, the above test can be based on a "classifier" representation of the applied policy along the line discussed in Section 2.2.1.

2. **PN-based policy representation**: The "behavioral filter" represented by the logical control policy is automatically imposed on the dynamics of the RAS-modeling PN through the superimposition to this net of additional structure representing the policy logic. In other words, after the RAS is modeled as a GSPN, some additional places with connections to the existing transitions, called *supervisors*, are added to impose the constraints that guarantee the liveness of the augmented net. For instance, if the liveness-enforcing constraints can be represented in the form of a set of linear inequalities $\sum_i a_i x_i \leq b$, where

[4]Note that any $t \in \mathcal{E}_t(M)$ models only the processing of the job instances, not any resource allocation behavior. Hence, no logical control is necessary for these timed transitions.

x_i are numbers of job instances in some processing stages in the RAS, and a_i and b are integers, then the constraints can be modeled by the addition of some supervisors, one for each constraint. Each of these supervisors enforces the corresponding linear constraint by establishing an invariant in the net dynamics similar to the invariants that are defined by the resource places. In fact, these supervisors can be perceived as additional "fictitious" resource types for the original net. The reader is referred to [37, 72, 47] for the corresponding analysis and the specific methods for the construction of the supervisors from: (i) the corresponding linear constraint, and (ii) the structure and the initial marking of the "plant" PN.

Policies and policy spaces When the deadlock avoidance policy (DAP) is integrated to the GSPN via one-step look-ahead control, the notions of policies and policy spaces can be introduced for the formalization of the imposed restriction on the behavior of the GSPN. More specifically, a *(stationary) policy* is a composite rule typically containing two tiers: the first tier is a disabling rule, which specifies the untimed transitions $t \in \mathcal{E}_u(M)$ to be disabled at each vanishing marking $M \in \mathcal{R}$; the second tier comprises the probability distributions, or the "random switches", that must be applied to all those vanishing markings with more than one untimed transitions enabled after the application of the disabling rule.[5] A policy with such a structure will be denoted by Z, and it is defined by the set of values of all the random switches that are employed by it. More formally,

$$Z \equiv \{Z_M \in \mathbb{R}_{0+}^{|\mathcal{T}_u|} : \ M \in \mathcal{R}_{\mathcal{V}}\} \tag{2}$$

[5] A policy can be non-stationary, if the composite rule depends also on the timing of the various decision points, besides the vanishing markings. But in this research context, only stationary policies will be considered. And the term "stationary" will be omitted in the sequel.

where the vectors Z_M satisfy the following additional constraints:

$$\sum_{t \in \mathcal{T}_u} Z_M[t] \leq 1, \ \forall M \in \mathcal{R}_{\mathcal{V}} \ \wedge \ Z_M[t] = 0, \ \forall M \in \mathcal{R}_{\mathcal{V}}, \ \forall t \notin \mathcal{E}_u(M) \tag{3}$$

A *policy space* Π is a set of policies that satisfy some constraints. The largest policy space for a GSPN is the *full policy space* Π_0, which contains any policies satisfying the conditions specified in (3). The inequality appearing in (3) enables the modeling of deliberate idleness. On the other hand, the one-step look-ahead DAP discussed previously applies the following additional constraint:

$$Z_M[t] = 0, \ \forall M \in \mathcal{R}_{\mathcal{V}}, \ \forall t \in \mathcal{E}_u(M) \text{ s.t. } tr(M,t) \text{ is inadmissible} \tag{4}$$

The restriction of the policy space Π_0 through the constraint of Equation (4) defines the policy space $\Pi_1 \subseteq \Pi_0$. Clearly, every policy Z in a policy space Π_1 which is defined by a correct DAP, is a deadlock and restricted deadlock-free policy.

Another constraint is applied on the policy space Π_1 to prevent higher priorities of the timed transitions over the untimed transitions (i.e., deliberate idleness other than that enforced by the applied DAP):

$$\sum_{t \in \mathcal{T}_u} Z_M[t] = 1, \ \forall M \in \mathcal{R}_{\mathcal{V}} \text{ s.t. } \exists t \in \mathcal{E}_u(M), tr(M,t) \text{ is admissible} \tag{5}$$

The policy space that results from Π_1 through the addition of constraint (5) will be denoted by $\Pi_2 \subseteq \Pi_1$. Although this constraint enforces the GSPN convention that untimed transitions have higher priorities than timed transitions, the reduction on the policy space that it incurs may lead to a potential sub-optimality of the new policy space [38]. However, in Section 3.5 we show that this issue can be easily addressed by properly augmenting the underlying GSPN structure.

The concept of *policy-space-conditioning* can be defined for the set of enabled untimed transitions, the feasible sequences, the reachable markings and the state space. More specifically, for a marking M, the set of Π-enabled untimed transitions

is defined as:

$$\mathcal{E}_u^{\Pi}(M) = \left\{t | \exists Z \in \Pi, \text{ s.t. } Z_M[t] > 0\right\}$$

Then, accordingly, a Π-feasible firing sequence $\sigma = t_1 t_2 \ldots t_l \in \mathcal{T}_u^*$ at marking M is a firing sequence such that $t_1 \in \mathcal{E}_u^{\Pi}(M)$, and for the ith transition in the sequence, $i = 2, \ldots, l$, $t_i \in \mathcal{E}_u^{\Pi}(tr(M, t_1 \ldots t_{i-1}))$. Furthermore, a marking M' is Π-reachable from M if there exists a Π-feasible firing sequence $\sigma \in \mathcal{T}_u^*$ at marking M such that $M \xrightarrow{\sigma} M'$. Finally, the notations $\mathcal{R}^{\Pi}$, $\mathcal{R}_{\mathcal{V}}^{\Pi}$ and $\mathcal{R}_{\mathcal{T}}^{\Pi}$ denote the conditional state spaces of all reachable markings, vanishing markings and tangible markings, respectively. Note that for a given policy space Π, due to the application of disabling rules for some untimed transitions, it is possible that some unconditional *vanishing* markings become Π-conditional tangible markings. Yet, it can be verified that $\forall \Pi \subseteq \Pi_0$, $\mathcal{R}_{\mathcal{V}}^{\Pi} \subseteq \mathcal{R}_{\mathcal{V}}$. When there is no superscript, the notation means "unconditional" and is under the setting of the full policy space Π_0.

In the sequel, most of the sets of enabled untimed transitions, the feasible firing sequences, the reachability from one marking to another, and the state spaces involved are Π_2-conditional, where Π_2 is the deadlock-free and non-idling policy space.

3.1.2 Some additional features of the RAS-modeling GSPN

To simplify the further analysis, it is convenient to confirm that the Conditions 1, 2 and 3 in Section 2.3.3 are satisfied in the considered class of GSPNs. As a first step, note that the total number of job instances is bounded, because (i) the total number of processing stages Δ of a RAS Φ is finite, and (ii) the total number of job instances at each processing stage is finite since in item 4 of Definition 1 it is assumed that at least one unit of at least one type of resource is required by any given process stage. Then, it can be further inferred that:

Proposition 1 *The RAS-modeling GSPN is bounded. Furthermore, its state space is finite, i.e.,* $|\mathcal{R}(\mathcal{GSPN}(\Phi))| < \infty$.

The second step seeks to establish the ergodicity of the stochastic process representing the timed dynamics of the GSPN, i.e., the satisfaction of Condition 2. As long as the GSPN is logically controlled by applying a correct DAP, it is reversible. In other words, the conditional reachability graph of the GSPN is strongly connected with respect to the policy space Π_1 or Π_2. Furthermore, for every policy Z from these two policy spaces that retains the corresponding conditional reachability graph as the STD of the induced SMP, we can infer that the EMC of this SMP is ergodic. Such policies Z can be obtained by employing random switches that assign a positive lower bound to the selection probability of each enabled untimed transition. In this way, we obtain the policy space Π_3 that is characterized by the addition of the following constraint to the policy space Π_2:

$$Z_M[t] \geq \delta / |\mathcal{E}_u^{\Pi_2}(M)|,\ \forall M \in \mathcal{R}_{\mathcal{V}}^{\Pi_2},\ \forall t \in \mathcal{E}_u^{\Pi_2}(M);\ 0 < \delta < 1 \tag{6}$$

In Equation (6), the parameter $\delta \in (0, 1)$ can be seen as a randomization factor. And the addition of the above constraint can be seen as the application of an *exploration mechanism* on the whole state space.

The next proposition follows directly from the above developments:

Proposition 2 *Consider a RAS-modeling GSPN and a policy $Z \in \Pi_3$. Then, the EMC of the corresponding SMP that represents the timed dynamics of the GSPN, is ergodic.*

The remaining step is the confirmation of Condition 3, which is given by Proposition 3:

Proposition 3 *The RAS-modeling GSPN, i.e., $\mathcal{GSPN}(\Phi)$, satisfies Condition 3. In other words, for any vanishing marking $M \in \mathcal{R}_{\mathcal{V}}(\mathcal{GSPN}(\Phi))$, there does not exist a feasible non-empty transition firing sequence $\sigma \in \mathcal{T}_u^*$ such that $M = tr(M, \sigma)$.*

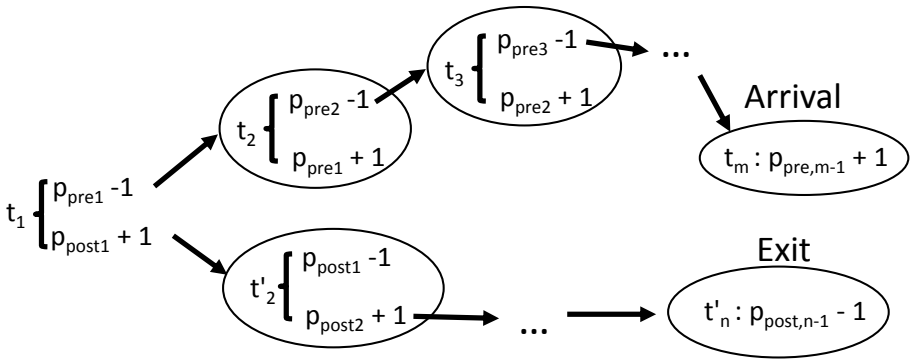


Figure 3.3: The set of transitions that "offset" the changes on the marking M that are incurred by the firing of transition t_1

Proof: The validity of this proposition mainly lies on the following two assumptions: (i) the no-revisit assumption for the sequential logic that is defined in item 3 of Definition 1, and (ii) the non-zero processing time assumption for any possible realization path of each process type (Assumption 3 in Section 2.1).

For the sake of contradiction, consider a RAS-modeling GSPN with a vanishing marking $M \in \mathcal{R}_{\mathcal{V}}$ and a feasible non-empty firing sequence $\sigma \in \mathcal{T}_u^*$ such that $M = tr(M, \sigma)$. Then, since σ is non-empty, there is a first transition t_1 in σ. According to the semantics of the RAS-modeling GSPN, the firing of a transition implies the stage advancement of one job instance j. Let us further assume that the job instance j is of process type Γ_j.

Without loss of generality, suppose that transition t_1 corresponds to the advancement of job instance j from process place p_{jk} to process place $p_{jk'}$ (the corresponding argument is similar for the remaining cases). Then, the firing of t_1 consumes one token at place p_{jk} and releases one token at place $p_{jk'}$. Since, by the no-revisit assumption of item 3 of Definition 1, the GSPN subnet modeling the process type Γ_j is acyclic, the considered sequence σ must contain two transition subsequences σ_1 and σ_2 that respectively replenish place p_{jk} with one token and abduct the extra token in place $p_{jk'}$ out of the network. Furthermore, the combined sequence $\sigma_1 t \sigma_2$ corresponds

to a complete realization path for process type Γ_j (c.f. Figure 3.3 for a visualization of this argument). Hence, by Assumption 3 in Section 2.1, σ must contain a timed transition, which contradicts the working assumption.

3.2 Specialization of the GSPN-based modeling of the RAS performance control problem to the CRL scheduling problem

The RAS performance control problem has been customized to a CRL scheduling problem in Section 2.4. And in this section, this line of customization will be further extended by constructing the corresponding GSPN model from a CRL configuration.

GSPN-based modeling of the considered CRL We start by noticing that the general procedures of Section 3.1 are still applicable, since the CRL operations are modeled as a RAS $\Phi(CRL)$. Next, consider a CRL stage J_j, which is modeled by the three processing stages of $\Phi(CRL)$: $\Delta_{j\mathrm{w}}$, $\Delta_{j\mathrm{p}}$ and $\Delta_{j\mathrm{b}}$. Note that stages $\Delta_{j\mathrm{w}}$ and $\Delta_{j\mathrm{b}}$ have zero processing time and each of them can be modeled by one process place. On the other hand, stage $\Delta_{j\mathrm{p}}$ has exponentially distributed processing time (or some more generally distributed processing time with positive support), so it should be modeled by two process places and an untimed transition (or resp., a subnet approximating the general distribution). All these four places can be connected with necessary untimed transitions that model the status change of the job instances. Finally, the resource places that model the availabilities of the server and the buffer slots of the different workstations can be connected to those untimed transitions according to the resource requirement function. The GSPN structure that models the CRL stage J_j is illustrated in the left side of Figure 3.4.

However, a simplification can be applied due to the special structure of the CRL. Consider the second process place that models the RAS stage $\Delta_{j\mathrm{p}}$. It models the blocking effect: a job finishes processing but it is blocked and cannot release the

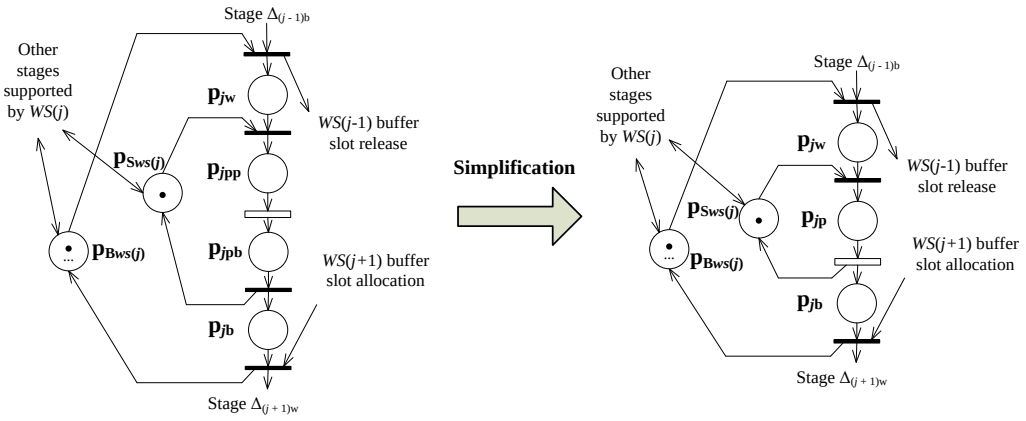


Figure 3.4: The GSPN model of a CRL stage J_j, before and after simplification

allocated resource. However, the blocking cannot happen here, because its next stage $\Delta_{j\mathrm{b}}$ does not require the allocation of new resources and any jobs finishing processing at $\Delta_{j\mathrm{p}}$ can immediately proceed to $\Delta_{j\mathrm{b}}$. This conclusion comes from the fact that the processing of CRL happens in situ, and the server will not be occupied by the job that finishes processing. As a result, the second place that models the stage $\Delta_{j\mathrm{p}}$ can be eliminated, and its input and output transitions can be combined. The simplification is depicted in the right side of Figure 3.4. Thanks to this simplification, there is a one-to-one relationship between the processing stages of the RAS $\Phi(CRL)$ and the process places of its GSPN model. And this relationship also defines the one-to-one relationship between the states of $\Phi(CRL)$ and the markings of the GSPN.

Finally, the reward function $\hat{r}$ on the set of tangible markings M of the GSPN built from the CRL model can be defined as

$$\hat{r}(M) = \begin{cases} r(t_{n\mathrm{p}}) & \text{if } t_{n\mathrm{p}} \in \mathcal{E}_t(M) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

In Equation (7), $t_{n\mathrm{p}}$ is the timed transition that models the processing of the last CRL stage. According to Assumption 4, the timing distribution for this last processing stage is exponential in the considered class of CRLs.

Deadlock avoidance in the CRL and the GSPN representation of the corresponding policy When dealing with deadlock avoidance problems in the CRL context, the simplified RAS model $\hat{\Phi}(CRL)$ can be used for analysis purposes. After the offline part of the logical control problem is solved, i.e., a classifier that implements the applied DAP becomes available, the classifier can be tailored to the state space of the complete RAS model, $\Phi(CRL)$, through the existing mapping between the state spaces of the two RAS models (c.f. Section 2.4).

As already discussed in this work, the employed DAP is enforced through the one-step look-ahead method that tests the admissibility of an enabled transition by running the resultant state through the constructed classifier. A special case for such tests is the deadlock-detection test for RAS which have no deadlock-free unsafe states. For this type of RAS, if the test gives a result of no deadlock, then it also implies the state safety. Since $\hat{\Phi}(CRL)$ is a linear and single-unit RAS, it contains, indeed, instantiations that possess no deadlock-free unsafe states. More specifically, Section 1 of Chapter 3 of [86] provides two theorems (Theorems 3.2 and 3.5) that characterize two types of single-unit RAS where no deadlock-free unsafe state exists. Both of these theorems are applicable to the linear and single-unit RAS, to which the class of the $\Phi(CRL)$ belongs. The theorems in [86] can be restated in the context of the simplified RAS $\hat{\Phi}(CRL)$ as follows:

Theorem 1 *[88, 32] A simplified RAS $\hat{\Phi}(CRL)$, that abstracts the buffer allocation in a given CRL, has no deadlock-free unsafe states, if every workstation i in the underlying CRL satisfies one of the following four conditions:*

(i). its number of buffer slots $B_i \geq 2$;

(ii). there is only one CRL stage j such that $\mathcal{WS}(j) = i$.

(iii). $\forall j < j'$ such that $\mathcal{WS}(j) = \mathcal{WS}(j') = i$, either j' is the final stage n, or $\mathcal{WS}(j+1) = \mathcal{WS}(j'+1)$;

(iv). $\forall j < j'$ *such that* $\mathcal{WS}(j) = \mathcal{WS}(j') = i$, *either* $j = 1$, *or* $\mathcal{WS}(j-1) = \mathcal{WS}(j'-1)$.

Theorem 2 *[58] A simplified RAS* $\hat{\Phi}(CRL)$*, that abstracts the buffer allocation in a given CRL, has no deadlock-free unsafe states, if there exists an indexing of the line workstations, such that every CRL stage* j *satisfies one of the following three conditions:*

(i). j *is the last stage* n *or the next last stage* $n-1$*;*

(ii). $j \leq n-2$ *and* $\mathcal{WS}(j+1) = \mathcal{WS}(j) + 1$*;*

(iii). $j \leq n-2$, $\mathcal{WS}(j+1) < \mathcal{WS}(j)$ *and* $\mathcal{WS}(j+2) > \mathcal{WS}(j+1)$.

The conditions of these two theorems are quite common in most applications of CRL. Furthermore, if a CRL has no deadlock-free unsafe states, then, the maximally permissive DAP can be represented by a linear classifier. To see this, consider any partial deadlock state s_{d} which is reachable from a safe state s_{s} by advancing a job instance to its next CRL stage or by loading a new job instance to the CRL. According to Definition 2, at state s_{d}, there exists a set of job instances $\mathcal{J}^{\mathrm{d}}(s_{\mathrm{d}})$, such that each of them holds one buffer slot and prevents the advancement of other job instances in $\mathcal{J}^{\mathrm{d}}(s_{\mathrm{d}})$. Furthermore, since RAS $\hat{\Phi}(CRL)$ concerns only the allocation of the buffer slots of the line workstations, it further follows that all the buffer slots requested by the jobs in $\mathcal{J}^{\mathrm{d}}(s_{\mathrm{d}})$ must correspond to buffers allocated to capacity. Hence, the considered deadlock can be prevented in a maximally permissive manner by the enforcement of the following inequality upon the state $\hat{s}$ of the simplified RAS $\hat{\Phi}(CRL)$:

$$\sum_{j \in \mathcal{J}^{\mathrm{d}}(s_{\mathrm{d}})} \hat{s}[j] \leq \sum_{i \in \mathcal{WS}^{\mathrm{d}}(s_{\mathrm{d}})} B_i - 1 \tag{8}$$

In Equation (8), $\mathcal{WS}^{\mathrm{d}}(s_{\mathrm{d}})$ is the set of all the workstations involved in the partial deadlock, i.e., $\mathcal{WS}^{\mathrm{d}}(s_{\mathrm{d}}) \equiv \{i : \exists j \in \mathcal{J}^{\mathrm{d}}(s_{\mathrm{d}}) \text{ s.t. } \mathcal{WS}(j) = i\}$.

The complete development of the constraints (8) requires the enumeration of all the reachable deadlocks, and falls into the category of the classifier-construction problem that was discussed in Section 2.2.1.

Example 1 revisited The GSPN model of the RAS $\Phi(CRL)$ constructed at the end of Chapter 1 is depicted in Figure 3.5, and the relevant semantics, modeling the corresponding process plan and the resource requirement function, are shown in Table 3.1. In particular, the figure and the table explain very clearly how the transitions that model the stage advancement of the job instances, are connected to the resource places, to model the resource allocation and release. Table 3.1 also provides the one-to-one relationship between the GSPN process places and the $\Phi(CRL)$ processing stages.

In Figure 3.5, the place p_{11} and its affiliated arcs are depicted in gray color and dashed lines. This place has not been included in Table 3.1 because it is a "supervisor" place that imposes the maximally permissive DAP for this CRL. More specifically, it can be easily checked that the simplified RAS $\hat{\Phi}(CRL)$ of this example satisfies the conditions of Theorem 1.[6] Therefore, this CRL does not possess any deadlock-free unsafe states, and the safety region can be characterized with linear constraints. Furthermore, it can be seen that for the simplified RAS $\hat{\Phi}(CRL)$, the only possible deadlock state is $\hat{s}_{\mathrm{d}} = (2\ 2\ 0)^T$ and the corresponding constraint of (8) takes the form:

$$\hat{s}[1] + \hat{s}[2] \leq 3$$

This constraint can be mapped to the complete RAS $\Phi(CRL)$ state s as follows:

$$s[1\mathrm{p}] + s[1\mathrm{b}] + s[2\mathrm{w}] + s[2\mathrm{p}] + s[2\mathrm{b}] \leq 3$$

[6] We remind the reader that the CRL contains two workstations and each of them has two buffer slots, which satisfies condition (i) of Theorem 1. In fact, this CRL also satisfies the condition of Theorem 2, as it supports the route among workstations (WS): $WS_1 \rightarrow WS_2 \rightarrow WS_1$. Of course, only one of these two theorems is enough for the conclusion.

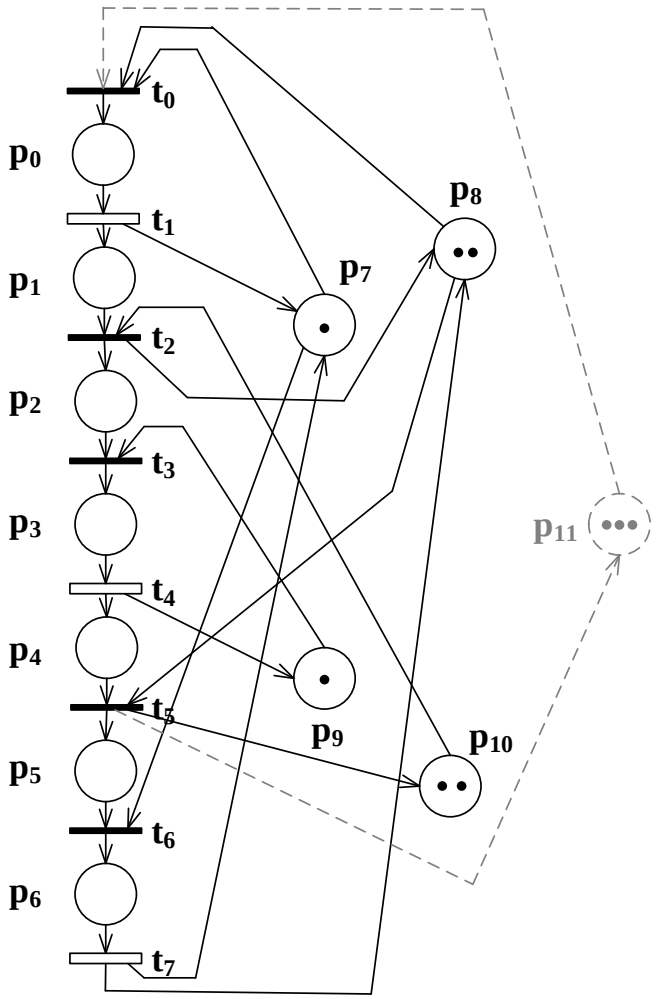


Figure 3.5: The GSPN model for the CRL scheduling problem in Example 1

Table 3.1: The operational semantics encoded by the GSPN in Figure 3.5

Process place	Description of the job status and the mapping to $\Phi(CRL)$ stages
p_0 (resp., p_3, p_6)	Being processed at workstation $\mathcal{WS}(1)$ (resp., $\mathcal{WS}(2)$, $\mathcal{WS}(3)$) $\Phi(CRL)$ stages: $\Delta_{1\mathrm{p}}$ (resp., $\Delta_{2\mathrm{p}}$, $\Delta_{3\mathrm{p}}$)
p_1 (resp., p_4)	Having completed processing at workstation $\mathcal{WS}(1)$ (resp., $\mathcal{WS}(2)$) and waiting for the advancement to the workstation required by the next stage $\Phi(CRL)$ stages: $\Delta_{1\mathrm{b}}$ (resp., $\Delta_{2\mathrm{b}}$)
p_2 (resp., p_5)	Waiting for processing at workstation $\mathcal{WS}(2)$ (resp., $\mathcal{WS}(3)$) $\Phi(CRL)$ stages: $\Delta_{2\mathrm{w}}$ (resp., $\Delta_{3\mathrm{w}}$)
Resource place	**Description**
p_7 (resp., p_9)	Server availability at workstation 1 (resp., 2)
p_8 (resp., p_{10})	Buffer slot availability at workstation 1 (resp., 2)
Transition	**Description of the actions that change job status**
t_0	Load and start processing a new job at workstation $\mathcal{WS}(1)$; Allocate the corresponding buffer slot and server
t_1, (reps., t_4)	Complete processing a job at workstation $\mathcal{WS}(1)$ (resp., $\mathcal{WS}(2)$); Release the corresponding server
t_2, (reps., t_5)	Advance a completed job at workstation $\mathcal{WS}(1)$ (resp., $\mathcal{WS}(2)$) to the workstation required by the next stage; Allocate and release the relevant buffer slots
t_3 (resp., t_6)	Start processing a job at workstation $\mathcal{WS}(2)$ (resp., $\mathcal{WS}(3)$); Allocate the corresponding server
t_7	Complete processing a job at workstation $\mathcal{WS}(3)$ and unload the job from the system, Release the corresponding server and buffer slot

Then, according to the one-to-one relationship between the $\Phi(CRL)$ stages and the GSPN process places, the constraint can be represented as a marking inequality

$$M[p_0] + M[p_1] + M[p_2] + M[p_3] + M[p_4] \leq 3$$

Finally, according to the theory of [72, 47], this marking inequality can be enforced by the addition of the supervisor place p_{11} as depicted in Figure 3.5.

As mentioned before, the supervisor places are not necessary if the employed DAP is implemented through a one-step look-ahead scheme. In this example, since the considered CRL satisfies the condition of Theorem 1, there is no deadlock-free unsafe state in $\hat{\Phi}(CRL)$. Hence, the test on whether an untimed transition t is admissible

at a vanishing marking M can be the test of whether the corresponding state $\hat{s}$ of $\hat{\Phi}(CRL)$ which is obtained from the marking $tr(M,t)$, is deadlocking.

3.3 The performance optimization of the RAS-modeling GSPN and its mathematical programming formulation

After the GSPN that models the logically controlled dynamics of the RAS is established, the remaining tasks are (i) the construction of the connection between the RAS scheduling/performance-oriented control objective and the performance optimization of the GSPN reward model, and (ii) the solution of the GSPN performance optimization problem. The first step essentially associates the reward rates to the tangible markings of the GSPN, according to the optimization objective and the application context of the underlying RAS. In this document, only throughput maximization is discussed. However, the methodological framework is compatible for any RAS performance optimization objectives that can be modeled as reward rates of tangible markings of the GSPN.

For the objective of throughput maximization, an indicator reward function $I_r(\cdot)$ is defined on the set of timed transitions: the function $I_r(t)$ returns one if the firing of the timed transition t models the completion of a last non-zero-time processing stage of some process type, since the firing of such a transition guarantees one unit of output from the RAS in zero time; and the function returns zero for any other timed transitions. This instantaneous reward subsequently can be changed to the form of a reward rate on any tangible marking M, denoted as the reward rate $\hat{r}(M)$:

$$\hat{r}(M) \equiv \sum_{t \in \mathcal{E}_t(M)} I_r(t) r(t)$$

where r is the mapping that defines the firing rates of the timed transitions (c.f. Section 2.3.2).

The performance optimization for the GSPN reward model was first addressed in [59], which gives the procedures for building a mathematical programming (MP)

formulation. In this document the procedures will be described in more detail.

After the GSPN is abstracted from a fully defined RAS, the set of decision variables are the vectors Z_M that can specify a policy Z as discussed in Section 3.1.1. When a correct DAP (including the maximally permissive DAP) is applied, no deliberate idleness is allowed, and the exploration mechanism of Equation (6) is applied, the policy Z is confined to the policy space Π_3, and the decision variables in Z_M can be restricted only to $M \in \mathcal{R}_{\mathcal{V}}^{\Pi_3} : |\mathcal{E}_u^{\Pi_3}(M)| \geq 2$, since the values of other vectors in Z have already been fixed by the constraints that define $\Pi_2 \supseteq \Pi_3$.[7]

Also, as mentioned in the definition of the policy space Π_3 in Section 3.1.1, the components of each Z_M are lower bounded by a positive scalar that is determined by the policy-space-defining parameter δ, in an effort to establish the ergodicity of the EMC of the underlying SMP. The transition rates of the underlying CTMC $\mathcal{M}$ corresponding to the GSPN can be obtained through the procedures illustrated at the end of Section 2.3.3. Therefore, the infinitesimal generator matrix $Q \in \mathbb{R}^{|\mathcal{R}_{\mathcal{T}}^{\Pi_2}| \times |\mathcal{R}_{\mathcal{T}}^{\Pi_2}|}$ of the CTMC $\mathcal{M}$ can be expressed as a matrix depending on Z, i.e., $Q(Z)$. Furthermore, a vector of auxiliary variables $\pi \in \mathbb{R}_{0+}^{|\mathcal{R}_{\mathcal{T}}^{\Pi_2}|}$ will denote the steady-state distribution of the CTMC $\mathcal{M}$. In this way, the problem of the performance optimization for the considered GSPN can be formulated as [59]:

$$\text{maximize} \quad \eta = \pi^T \cdot \hat{r} \tag{9}$$

$$\text{subject to} \quad \pi^T Q(Z) = \mathbf{0} \tag{10}$$

$$\pi^T \mathbf{1} = 1 \tag{11}$$

$$\sum_t Z_M[t] = 1 \qquad \forall M \in \mathcal{R}_{\mathcal{V}}^{\Pi_2} : \ |\mathcal{E}_u^{\Pi_2}(M)| \geq 2 \tag{12}$$

$$Z_M[t] \geq \frac{\delta}{|\mathcal{E}_u^{\Pi_2}(M)|} \qquad \forall M \in \mathcal{R}_{\mathcal{V}}^{\Pi_2} : \ |\mathcal{E}_u^{\Pi_2}(M)| \geq 2; \ \forall t \in \mathcal{E}_u^{\Pi_2}(M) \tag{13}$$

[7] Since the additional constraint (6) defining Π_3 from Π_2 does not disable any untimed transitions, $\mathcal{E}_u^{\Pi_2}(M) = \mathcal{E}_u^{\Pi_3}(M)$, for any $M \in \mathcal{R}$. Therefore, the state spaces defined by Π_2 and Π_3 are the same. In the sequel, all the concepts related to the underlying state space or reachability will be denoted as Π_2-conditional instead of Π_3-conditional.

Equation (9) expresses the objective of maximizing the steady-state average reward.[8] The system of equations (10)–(11) in the set of constraints defines the steady-state distribution. Finally, the inequality constraints (12)–(13) ensure that the pricing of the decision variables specifies legal probability distributions with the desired degree of randomization.

The problem formulation of Equations (9)–(13) can be further simplified with the removal of all the equality constraints in the formulation and the reduction of the corresponding decision variables. More specifically, let vector $\bar{\zeta}$ be the set of decision variables after removing the equality constraints, and define an index-searching function $idx : \mathcal{R}_{\mathcal{V}}^{\Pi_2} \times \mathcal{T}_u \mapsto \mathbb{Z}_+ \cup \{\texttt{NULL}\}$ as follows: with input of a vanishing marking M and an untimed transition t, if $|\mathcal{E}_u^{\Pi_2}(M)| \geq 2$ and t is not the last element in $\mathcal{E}_u^{\Pi_2}(M)$ according to the lexicographic order that is based on the natural numbering of the transition set $\mathcal{T}$, then $idx(M,t)$ returns the index number i in the vector $\bar{\zeta}$ such that $\bar{\zeta}[i]$ is the probability that t is fired at M, i.e., $Z_M[t] \equiv \bar{\zeta}[idx(M,t)]$; otherwise, $idx(M,t)$ returns NULL. Also suppose that the objective function can be expressed as $\eta = \eta(\bar{\zeta})$. Then, the MP formulation of Equations (9)–(13) can be re-written as

$$\text{maximize} \quad \eta(\bar{\zeta}) \tag{14}$$

$$\text{subject to} \quad \sum_{t:idx(M,t)\neq\texttt{NULL}} \bar{\zeta}[t] \leq 1 - \frac{\delta}{|\mathcal{E}_u^{\Pi_2}(M)|} \qquad \forall M \in \mathcal{R}_{\mathcal{V}}^{\Pi_2} : |\mathcal{E}_u^{\Pi_2}(M)| \geq 2 \tag{15}$$

$$\bar{\zeta}[idx(M,t)] \geq \frac{\delta}{|\mathcal{E}_u^{\Pi_2}(M)|} \quad \forall M \in \mathcal{R}_{\mathcal{V}}^{\Pi_2},\ \forall t \in \mathcal{T}_u : idx(M,t) \neq \texttt{NULL} \tag{16}$$

The input to the formulation of Equations (14)–(16) includes: the PN structure $\mathcal{N}$, the initial marking M_0, the partition of the transitions $\mathcal{T}_u$ and $\mathcal{T}_t$, the function r that defines the transition rates of the timed transitions, and the vector or function $\hat{r}$ that defines the reward rate. The solution of this formulation consists of a vector $\bar{\zeta}^*$ that specifies the optimal pricing of the selection probabilities in the policy space

[8] Similar to the steady-state distribution vector π and the infinitesimal generator Q, the reward vector $\hat{r}$ is a bit different from its original definition in Section 2.3.3. In this formulation, $\hat{r}$ is defined on the set of Π_2-conditional tangible markings $\mathcal{R}_{\mathcal{T}}^{\Pi_2}$, not the original set of tangible markings $\mathcal{R}_{\mathcal{T}}$.

Π_3, and the value of the objective function $\eta^* = \eta(\bar{\zeta}^*)$. As $\delta \downarrow 0$, Π_3 converges to the deadlock-free and non-idling policy space Π_2, and thus, η^* converges to the maximum steady-state average reward of the GSPN and the RAS with no deliberate idleness.[9]

Uniformization The dependency of η on $\bar{\zeta}$ can be either resolved through the system of equations (10)–(11), or the steady-state distribution of the discrete-time Markov chain (DTMC) $\hat{\mathcal{M}}$ after the uniformization of the CTMC $\mathcal{M}$. Though the uniformization does not render easier the (direct) solution of the MP formulation of Equations (14)–(16), it is useful when this formulation is approached through the numerical methods coming from Markov-chain theory, such as the computing methods for Markov decision processes (MDPs), or simulation optimization. The uniformization can be applied in the following way:

1. Compute a rate r_{u}, which is an upper bound to the sum of the rates of all the simultaneously processed job instances in the underlying RAS plus a small positive scalar. This rate r_{u} can be adopted as the *uniformization rate*. For the special case of a CRL satisfying Assumption 4, the aforementioned upper bound can be set to $\max_i \sum_{j:\mathcal{WS}(j)=i} \mu_j$. The small positive scalar is used to give positive transition probabilities for the self-loop transitions, and guarantee the aperiodicity of this Markov chain, which is a desirable feature.

2. Change the reward rate vector $\hat{r}$ to the vector $\frac{\hat{r}}{r_{\mathrm{u}}}$. These rewards can be perceived as one-step immediate rewards in the operation of the uniformized Markov chain. Change the average reward η to the expected reward per transition in steady state: $\frac{\eta}{r_{\mathrm{u}}}$.

3. Generate the non-diagonal components of the transition probability matrix P of the DTMC $\hat{\mathcal{M}}$, by dividing by r_{u} the components at the same position in the

[9] The constraints (15) and (16) make the policy space Π_3 a proper subset of the policy space Π_2, as long as $\delta > 0$.

infinitesimal generator Q of the CTMC $\mathcal{M}$.

4. Generate the diagonal components of P of the DTMC $\hat{\mathcal{M}}$, by subtracting from one the sum of the other components in the same row.

In the sequel, the notation $\hat{r}$ and η may have either their original meaning or the quantities that are divided by r_{u}, depending on the context. It is well known that the steady-state distribution π is not changed by the presented uniformization process [16].

Example 1 revisited We remind the reader that the GSPN model for this example CRL was constructed at the end of Section 3.2. At that point, the maximally permissive DAP was also implemented. Here we address the MP formulation for the performance optimization of the GSPN corresponding to this example.

The Π_2-conditional STD of the SMP corresponding to the GSPN of this example is presented in Figure 3.6. The token distribution in the process places of the net at each reachable marking, can be found in Table B.1 of Appendix B. In the depicted STD, the nodes model all the reachable markings. Among them, the single-circled nodes model the vanishing markings, while the double-circled nodes model the tangible markings. And the arcs from the tangible nodes model the enabled timed transitions, while the arcs from the vanishing nodes model the Π_2-enabled untimed transitions. The branching probabilities from the tangible markings are determined by the exponential race taking place among the timed transitions that are enabled at these markings, and are presented as labels on their corresponding arcs. However, the branching probabilities from the vanishing markings M depicted as gray nodes, which are "decision points", should be determined by the corresponding vectors Z_M of the random switches. In this example, there are 20 random switches with 27 total degrees of freedom for the policy space, which are modeled by 27 decision variables.

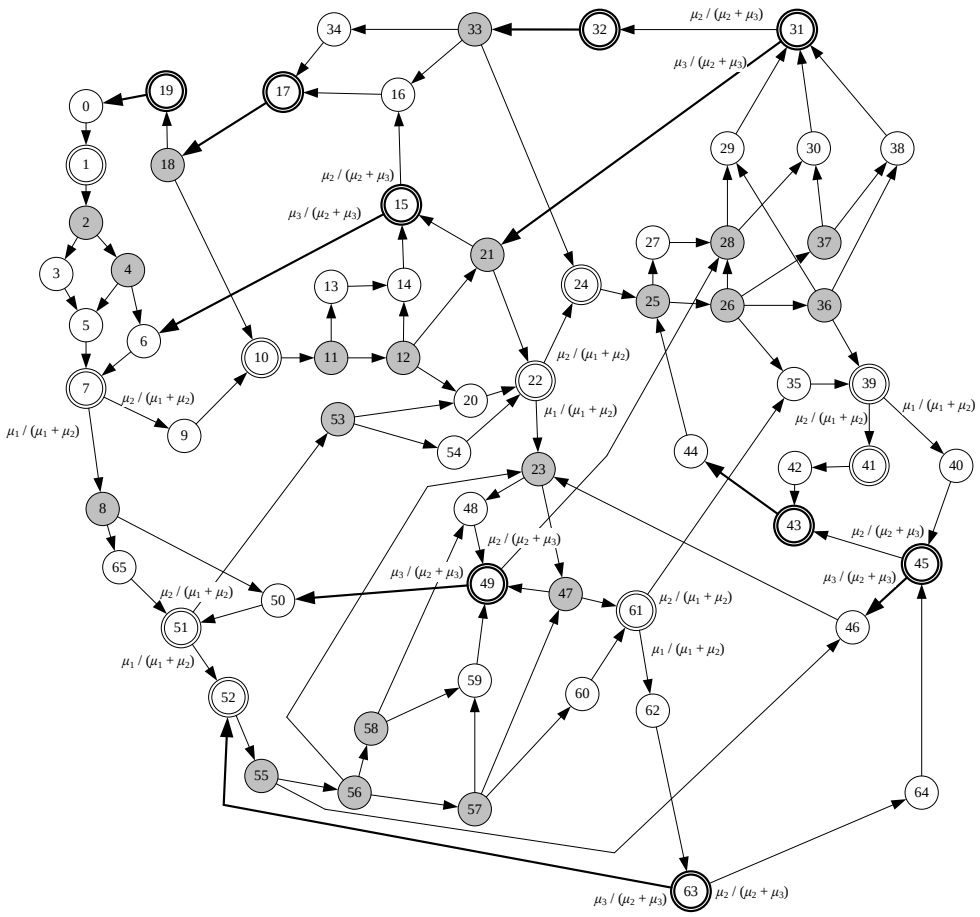


Figure 3.6: The Π_2-conditional state transition diagram of the underlying semi-Markov process for Example 1

Furthermore, the reward rates are defined on $M \in \mathcal{R}_{\mathcal{T}}$ as follows:

$$\hat{r}(M) = \begin{cases} \mu_3 & \text{if } t_7 \in \mathcal{E}_t(M) \\ 0 & \text{otherwise} \end{cases}$$

In Figure 3.6, the tangible markings with non-zero reward rates are depicted in thicker double-circled nodes, and the enabling of transition t_7 is depicted by thicker arrows. In fact, in any CRL scheduling problems that maximize the steady-state throughput, the tangible markings with positive reward rates also have equal reward rates.

Finally, the uniformization rate r_{u} is $\max\{\mu_1+\mu_3, \mu_2\}$ plus a small positive scalar. And then, the reward rates on the tangible markings where t_7 is enabled become the immediate reward $\frac{\mu_3}{r_{\mathrm{u}}}$.

In summary, the MP formulation for this example is controlling the branching probabilities at the gray nodes in order to maximize the sum of the steady-state probabilities corresponding to the thicker double-circled nodes. The reader may also observe that some decision points are redundant (e.g., the selection between markings 3 and 4 at marking 2, since either choice leads deterministically to tangible marking 7); the elimination of such redundancy and the formalization of this process will be the topic of Section 4.1 in the next chapter.

3.4 Complexity considerations

While the NP hardness of the deadlock avoidance problem for the sequential RAS considered in this work has been successfully addressed through the various methods that were discussed in the earlier parts of this document, there are challenging complexity issues that remain open when it comes to the performance optimization of these RAS. More specifically, it should be evident to the reader that the "size" of the MP formulation of (14)–(16) in terms of the employed decision variables and constraints, is super-polynomial with respect to $|\Phi|$. In this section, we take a closer look at this "super-polynomial" complexity and its implications.

If the state of the considered RAS is represented by the number of job instances at each stage, then the size of the whole state space, including the unsafe and possibly some infeasible states, should be the product of the number of possibilities at each stage, i.e.,

$$\left|\mathcal{R}^{\Pi_1} \cup \mathcal{R}_{unsafe} \cup \mathcal{R}_{infeasible}\right| = O\left(\prod_{j,k} \min_{i:A_{jk}[i]>0} \frac{C_i}{A_{jk}[i]}\right) \tag{17}$$

Equation (17) adopts the RAS notations rather than their GSPN counterparts. More specifically, i is the index number of the resource type, C_i is capacity of resource type i, and A_{jk} is the resource requirement vector for processing stage Δ_{jk} of process type Γ_j. The product in Equation (17) implies that the size of the whole state space is in a form of $|\Phi|^{|\Phi|}$. Hence, the size of the whole state space is super-polynomial with respect to $|\Phi|$. Furthermore, it is also true that the confinement of the system operation to the set of its safe states (and possibly also to its non-deliberately-idling states) leads to a new state space that is of the same order of magnitude with the original state space. Therefore, the numbers $|\mathcal{R}^{\Pi_1}|$ and $|\mathcal{R}^{\Pi_2}|$ are still super-polynomial with respect to $|\Phi|$. Such a super-polynomial complexity causes two types of difficulties in the solution of the MP formulation:

1. The number of vectors Z_M that are employed by any candidate policy Z is the number of Π_2-conditional vanishing markings which enable more than one untimed transitions. In the worst case, this number is polynomial with respect to $|\mathcal{R}^{\Pi_2}|$, and therefore, the number of vectors Z_M is super-polynomial with respect to $|\Phi|$. This fact can raise significant complications not only in the computation of an optimized policy Z, but also in the mere representation of this policy.

2. In order to evaluate the value of the objective function (9), the system of equations (10)–(11) must be solved. However, an explicit representation of the solution of this system of equations can also be intractable due to the explosive

size of the underlying state space.

These two issues are addressed in the rest of this document. In particular, Chapter 4 applies several steps to refine and approximate the solution space, in a way that reduces the dimension of the solution space to a manageable level. This chapter also introduces a mechanism that can explicitly control the representational complexity of the employed policy spaces. Chapter 5 addresses the second of the aforementioned problems by taking advantage of the fact that the objective function (9) constitutes an expectation with respect to the steady-state distribution π. Therefore, a simulation optimization algorithm is adopted to solve the problem.

3.5 A closing remark

There are some cases where the feature of deliberate idleness can be removed from the RAS operation without impacting the optimality of its performance. Some of these cases in the CRL context have been identified in [19] (c.f. Lemmas 3–9 in Section 4.2.2). However, there is no guarantee that the elimination of deliberate idleness will maintain, in general, the "performance potential" of the policy space [38]. As a result, the enforcement of the non-deliberately-idling constraint (5) in the policy space Π_2, might render this policy space sub-optimal with respect to the policy space Π_1.

However, next we show that deliberate idleness can be implemented by properly augmenting the net structure of the RAS-modeling GSPN, in a way that eventually we shall be able to address this element in the policy space Π_2 of the modified GSPN. The required structure includes (i) an untimed transition t_{idle} that models the decision for deliberate idleness, and (ii) a place p_{idle} that models the conflict between an untimed transition in the original net and the transition t_{idle}. More specifically, the place p_{idle} is contained in the pre-set $\bullet t$ of every untimed transition t, including the transition t_{idle}; it is also contained in the post-set $t\bullet$ of every timed or untimed transition t in the original net, i.e., $\mathcal{T}_u \cup \mathcal{T}_t \setminus \{t_{\text{idle}}\}$. In the initial marking M_0, which is vanishing, there

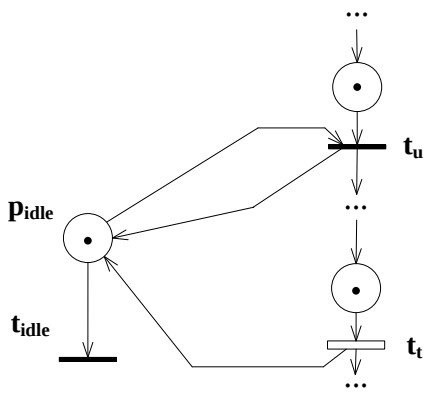


Figure 3.7: An illustration of the net structure that integrates the action of deliberate idleness in the RAS-modeling GSPN

is one token in the place p_{idle}. The above structure is illustrated in Figure 3.7, which depicts a RAS-modeling GSPN at a Π_1-reachable vanishing marking M_1. In the GSPN of this figure, only a part of the process-type modeling subnet is depicted, and the resource-modeling and DAP-implementation subnets are omitted. We suppose that $\mathcal{E}_u^{\Pi_1}(M_1) = \{t_{\text{idle}}, t_u\}$, and $\mathcal{E}_t(M_1) = \{t_t\}$. It is also true that $t_u \in \mathcal{E}_u^{\Pi_1}(tr(M_1, t_t))$.

At marking M_1, the firing of transition t_{idle} implies an action of deliberate idleness, since the untimed transition t_u is disabled by the removal of the token in the place p_{idle}, and the resultant marking $M_2 \equiv tr(M_1, t_{\text{idle}})$, which is essentially equal to M_1 except that $M_2[p_{\text{idle}}] = 0$, becomes a tangible marking. Once the enabled timed transition t_t is fired, a token is released to place p_{idle} and the untimed transition t_u is enabled again since $t_u \in \mathcal{E}_u^{\Pi_1}(tr(M_1, t_t))$.[10] On the other hand, if the enabled untimed transition t_u is fired first, then the token in the place p_{idle} is not consumed and the transition t_{idle} remains enabled; therefore, the resultant marking $M_3 \equiv tr(M_1, t_u)$ is vanishing with $\mathcal{E}_u^{\Pi_1}(M_3) = \{t_{\text{idle}}\}$. After the only enabled untimed transition t_{idle} is fired, the resultant marking $tr(M_3, t_{\text{idle}})$ is tangible. Finally, it is also important to notice that since the enabled timed transitions can be fired only one at a time, and all

[10]We remind the reader that the firing of a timed transition modeling the completion of a processing stage (or of a certain phase of a processing stage) does not involve the allocation of any new resource units.

enabled untimed transitions have precedence over timed ones, the place p_{idle} cannot hold more than one token at any time.

It should be clear from the above description that the introduction of the place p_{idle} and the transition t_{idle} as indicated in Figure 3.7, enables the handling of deliberate idleness through the specification of the random switches of the augmented GSPN by means of the MP formulation of Section 3.3. However, the proper functioning of this modeling scheme requires an additional deadlock-avoidance constraint:

$$Z_M[t_{\text{idle}}] = 0, \ \forall M \in \mathcal{R} \text{ s.t. } \mathcal{E}_t(M) = \emptyset \tag{18}$$

Otherwise, the firing of t_{idle} can disable all the transitions, and the GSPN gets in a deadlock state.

CHAPTER IV

CONTROLLING THE REPRESENTATIONAL COMPLEXITY

As discussed in Section 3.4, the RAS performance optimization problem that is considered in this work can become intractable due to the explosive increase of the size of the underlying RAS state space. And this intractability is manifested with respect to, both, the representational and the computational complexity of the target policies. This chapter seeks to address the issues related to the representational complexity of the target policies by introducing three important mechanisms: *refinement, restriction*, and *partial disaggregation* of the considered policy spaces. All these mechanisms seek to address the representational complexity of the considered optimization problem by adjusting the number of the primary decision variables in the corresponding MP formulation. The remaining intractability issue that pertains to the computational complexity of estimating the value of the objective function of Equation (14), is addressed through the simulation optimization techniques that are introduced in the next chapter.

4.1 A random-switch refinement process

We remind the reader that, in the GSPN context, a random switch is defined as a probability distribution that regulates the firing of a set of enabled untimed transitions at a vanishing marking. In the context of the performance optimization problem that is considered in this work, it is possible that the selection among some enabled untimed transitions at certain vanishing markings of the RAS-modeling GSPN does not impact the attained performance. The topic of this section is the identification and removal

of such "redundant" options in random switches. The presented ideas and methods were first introduced in [60]. This section adapts these earlier developments to the "policy space" structures that were introduced in Section 3.1.1. We provide a formal treatment of this material, starting from the next subsection.

However, before we delve into these developments, we also want to point out that, from a more conceptual standpoint, the line of the results that are presented in this section was motivated by concepts and results pertaining to the notions of "conflict" and "conflicting behaviors" in DES theory at large [94], and in PN theory, in particular [25]. Along these lines, we remind the reader that in DES theory two enabled events are said to be in conflict at a certain state, if the execution of one of them leads to the disablement of the other. Events that are not in conflict can be executed consecutively in any sequence, and without any need for arbitration among them. Furthermore, the work of [38] associates the lack of conflict in the dynamics of any given DES with the ability of a "greedy" policy that executes every event upon its enablement, to expedite the (repeated) execution of the various events that can take place in this system.

Based on these results and insights, it is tempting to try to control the number of variables that appear in various random switches of the pursued GSPN models, by avoiding to control transitions that are not in conflict. It turns out, however, that in the context of the performance optimization problems that are undertaken in this work, the aforementioned classical notion of conflict is not completely appropriate for supporting the suggested "thinning" of the employed random switches, and it must be replaced by a modified version. We shall introduce this new version, and we shall discuss its explanatory role for the structural results that are developed in this section, at a later point, as we progress through the corresponding developments.

4.1.1 Formalization of the random-switch refinement process

Since the objective that is undertaken in this section is the identification and removal of the "redundant" options in random switches, we need to define the employed notion of redundancy in a clear and pertinent manner.

For that, first we notice that according to the developments of Section 2.3.3, the performance criteria for the considered RAS optimization problem only depend on the CTMC $\mathcal{M}$ defined on the sub-space of the tangible markings of the RAS-modeling GSPN. In other words, it is the Markovian "macro-transitions" among the set of tangible markings that will decide the steady-state average reward of the underlying RAS. This remark motivates the following definition.

Definition 5 *Given a RAS-modeling GSPN, two policies Z and Z' are* performance-equivalent *if their corresponding CTMCs, calculated using the steps that are described at the end of Section 2.3.3, are the same.*

In the light of Definition 5, our objective is the construction of a subset of the original policy space Π, to be called a *refined* policy space and be denoted by $\hat{\Pi}$, such that the policy space $\hat{\Pi}$ contains a policy $\hat{Z}$ that (i) is performance-equivalent to some optimal policy $Z^* \in \Pi$ and (ii) admits a simpler representation than policy Z^*. The next two definitions formalize this requirement.

Definition 6 *In the context of this work, a policy space $\hat{\Pi} \subseteq \Pi$* refines *the original policy space Π if it is derived from Π through the addition of the constraint*

$$Z_M[t] = 0 \tag{19}$$

for some vanishing markings $M \in \mathcal{R}_{\mathcal{V}}^{\Pi}$ and some untimed transitions $t \in \mathcal{E}_u^{\Pi}(M)$.

Definition 7 *Given an original policy space Π, a refined policy space $\hat{\Pi} \subseteq \Pi$* maintains the performance potential *of Π, if there exist an optimal policy $Z^* \in \Pi$ and a policy $\hat{Z} \in \hat{\Pi}$ such that Z^* and $\hat{Z}$ are performance-equivalent.*

Remark The original policy space Π that is employed in the subsequent developments is the deadlock-free and non-deliberately-idling policy space Π_2. As a result, in the subsequent deliberations the refined policy space will be denoted by $\hat{\Pi}_2$. We restrict our attention to the policy space Π_2 for the following two reasons: (i) First, we remind the reader that the developments of Section 3.5 have established that it is possible to model deliberate idleness in the underlying RAS while working in the policy space Π_2, through the introduction to the original RAS-modeling GSPN of the additional structure that is depicted in Figure 3.7. Hence, there is no particular advantage in working with the policy space Π_1. (ii) On the other hand, the results that we establish in this section will hold only in an approximate sense in the policy space Π_3; we shall discuss the details of this approximation as we progress with the technical development of the corresponding material. $\square$

In view of the modeling objectives that have been set for this section, we are particularly interested in refined policy spaces $\hat{\Pi}_2$ that maintain the performance potential of the original policy space Π_2 and minimize the number of the free decision variables that are employed by the corresponding policies $\hat{Z}$. However, the effective computation of such a refined policy space requires a holistic view of the state space of the corresponding SMP, and therefore, it will be intractable for exactly the same reasons that render intractable the original RAS optimization problem. Hence, in the subsequent developments, we compromise with refined policy spaces $\hat{\Pi}_2$ that can be claimed "minimal" according to some more local criteria. The notion of this "minimality" is detailed in the next section.

4.1.2 Construction of a "minimal" refined policy space

The departing point for the developments that are presented in this section, is the realization that, in the context of the original policy space Π_2, the removal of a transition t from the set $\mathcal{E}_u^{\Pi_2}(M)$, at any given vanishing marking M, will not compromise

the performance potential of Π_2, as long as this removal will not impact the set of tangible markings that are Π_2-reachable from M. The next two definitions will help provide the necessary formal statements and develop rigorously the aforementioned result.

Definition 8 *The* Π-untimed reach *of a marking* M *of a RAS-modeling GSPN, denoted as* $\mathcal{UR}^{\Pi}(M)$*, is the set of markings* M' *that are reachable from* M *by firing some* Π*-feasible transition sequence in* $\mathcal{T}_u^*$*.*

Definition 9 *The* Π-untimed tangible (resp., vanishing) reach *of a marking* M *of a RAS-modeling GSPN, denoted as* $\mathcal{UR}_{\mathcal{T}}^{\Pi}(M)$ *(resp.,* $\mathcal{UR}_{\mathcal{V}}^{\Pi}(M)$*), is the subset of* Π*-conditional tangible (resp., vanishing) markings in the* Π*-untimed reach* $\mathcal{UR}^{\Pi}(M)$*, i.e.,* $\mathcal{UR}_{\mathcal{T}}^{\Pi}(M) = \mathcal{UR}^{\Pi}(M) \cap \mathcal{R}_{\mathcal{T}}^{\Pi}(M)$ *(resp.,* $\mathcal{UR}_{\mathcal{V}}^{\Pi}(M) = \mathcal{UR}^{\Pi}(M) \cap \mathcal{R}_{\mathcal{V}}^{\Pi}(M)$*).*

The notion of the untimed tangible reach of any given marking M is connected to the aforestated objective of developing refined policy spaces $\hat{\Pi}_2$ that will maintain the performance potential of the original policy space Π_2, through the following proposition:

Proposition 4 *Let* $\hat{\Pi}_2$ *be a refined policy space of* Π_2 *such that* [1]

$$\forall M \in \mathcal{R}^{\Pi_2}, \;\; \mathcal{UR}_{\mathcal{T}}^{\Pi_2}(M) \;=\; \mathcal{UR}_{\mathcal{T}}^{\hat{\Pi}_2}(M) \tag{20}$$

Then, $\hat{\Pi}_2$ *maintains the performance potential of the original policy space* Π_2*.*

Proof: We will construct an optimal policy $\hat{Z} \in \Pi_2$ and prove also that $\hat{Z} \in \hat{\Pi}_2$.

[1] Since the constraints imposed by the policy space $\hat{\Pi}_2$ are applicable to any markings in the full state space $\mathcal{R}^{\Pi_0}$ (or simply $\mathcal{R}$), the calculation of $\mathcal{UR}_{\mathcal{T}}^{\hat{\Pi}_2}(M)$ is possible even if $M \in \mathcal{R}^{\Pi_2} \setminus \mathcal{R}^{\hat{\Pi}_2}$.

For this, first we notice that the considered performance optimization problem, defined on the original policy space Π_2, can be modeled as an average-reward semi-Markov decision process defined on the state space $\mathcal{R}^{\Pi_2}$.[2] If the state M is a vanishing marking, then the set of available actions $\mathcal{A}(M) = \mathcal{E}_u^{\Pi_2}(M)$; once an action $t \in \mathcal{E}_u^{\Pi_2}(M)$ is taken, the system goes to the state $tr(M,t)$ with probability one. Otherwise, the state M is a tangible marking with the corresponding action set $\mathcal{A}(M)$ being the singleton "do nothing", and the transition probabilities to the next states being defined by the exponential race that takes place at this tangible marking. Then, thanks to the communicating structure that is implied by the policy space Π_2, there exists a (relative) value vector $h^* \in \mathbb{R}^{|\mathcal{R}^{\Pi_2}|}$ that solves the optimality equation [11] (c.f. Equation (5.43), pg. 312):

$$h^*[M] = \max_{a \in \mathcal{A}(M)} \left(G(M,a) - \eta^* \bar{\tau}_M(a) + \sum_{M' \in \mathcal{R}^{\Pi_2}} p(M, M'; a) h^*[M'] \right), \quad M \in \mathcal{R}_{\Pi_2} \tag{21}$$

In Equation (21), $G(M,a)$ is the expected immediate reward of the state-action pair (M,a); $\bar{\tau}_M(a)$ is the expected sojourn time at state M when action a is taken; η^* is the maximum steady-state average reward; and $p(M, M'; a)$ is the transition probability from state M to M' under action a.

Let us derive some further useful characteristics of the vector h^*. If M is a vanishing marking, then $G(M,a) = \bar{\tau}_M(a) = 0$, regardless of the action taken. Also, the action a is the firing of a Π_2-enabled untimed transition t at M. Thus, $P(M, M'; a) = I_{\{M' = tr(M,t)\}}$, where I_E denotes the indicator function of event E. But then, for any vanishing marking M:

$$h^*[M] = \max_{t \in \mathcal{E}_u^{\Pi_2}(M)} h^*[tr(M,t)] \tag{22}$$

[2]This Markov decision process is different from the one in Appendix A, since the model constructed here is on the state space of all the markings, and the stochastic process induced by any given stationary policy is a semi-Markov process rather than a Markov process.

Furthermore, a simple inductive argument on the untimed vanishing reach $\mathcal{UR}_{\mathcal{V}}^{\Pi_2}(M)$ of M will establish that

$$h^*[M] = \max_{M_{\mathcal{T}} \in \mathcal{UR}_{\mathcal{T}}^{\Pi_2}(M)} h^*[M_{\mathcal{T}}] \tag{23}$$

According to Equation (23), the h^* value of any vanishing marking $M \in \mathcal{R}_{\mathcal{V}}^{\Pi_2}$ is equal to the highest h^* value among the h^* values of the tangible markings in $\mathcal{UR}_{\mathcal{T}}^{\Pi_2}(M)$.

Then, we have enough information for the construction of the sought policy $\hat{Z} \in \Pi_2$. More specifically, for any vanishing marking $M \in \mathcal{R}_{\mathcal{V}}^{\Pi_2}$, we select the untimed transition $\hat{t}$ to be fired at M as any transition

$$\hat{t} \ \in \ \arg\max_{t \in \mathcal{E}_u^{\Pi_2}(M)} h^*[tr(M,t)] \ \cap \ \mathcal{E}_u^{\hat{\Pi}_2}(M) \tag{24}$$

where the vector h^* is the aforementioned solution of the optimality equation with respect to the original policy space Π_2.

The definition of such a policy is possible, thanks to (i) Equation (19) that defines the refining process providing the policy space $\hat{\Pi}_2$, and (ii) the presumed condition in Proposition 4, which guarantees the non-emptiness of the intersection in the right-hand-side of Equation (24). Furthermore, the policy $\hat{Z}$ specified by Equation (24) is an optimal policy in Π_2 since it is a greedy policy with respect to the vector h^*. Also, it is obvious that $\hat{Z} \in \hat{\Pi}_2$. Therefore, the policy space $\hat{\Pi}_2$ maintains the performance potential of Π_2. $\square$

Obviously, the refinement of the random switches that is attempted in this section pertains only to the vanishing markings of the underlying GSPN. For such a vanishing marking M, the right-hand-side of the condition of Proposition 4 can be equivalently expressed as $\bigcup_{t \in \mathcal{E}_u^{\hat{\Pi}_2}(M)} \mathcal{UR}_{\mathcal{T}}^{\hat{\Pi}_2}(tr(M,t))$, leading to the following restatement of this

condition:

$$\begin{aligned} \mathcal{UR}_{\mathcal{T}}^{\Pi_2}(M) &= \bigcup_{t \in \mathcal{E}_u^{\hat{\Pi}_2}(M)} \mathcal{UR}_{\mathcal{T}}^{\hat{\Pi}_2}(tr(M,t)) \\ &= \bigcup_{t \in \mathcal{E}_u^{\hat{\Pi}_2}(M)} \mathcal{UR}_{\mathcal{T}}^{\Pi_2}(tr(M,t)) \end{aligned} \tag{25}$$

An algorithm for the computation of the sought policy space $\hat{\Pi}_2$ The condition of Equation (25) provides the basis for the development of an algorithmic procedure for the construction of the sought policy space $\hat{\Pi}_2$. More specifically, next we shall develop an algorithm that, for any given vanishing marking M, will select a *minimal-cardinality* transition subset $\mathcal{E}_u^{\hat{\Pi}_2}(M) \subseteq \mathcal{E}_u^{\Pi_2}(M)$ that satisfies this condition. Then, in view of Proposition 4 and the above discussion, the constructed policy space $\hat{\Pi}_2$ will maintain the performance potential of the original policy space Π_2. We formalize this construction through the following definitions:

Definition 10 *Given a vanishing marking M of a RAS-modeling GSPN, a set of untimed transitions $\mathcal{E}$ will be characterized as* Π-irreducible *with respect to the considered vanishing marking M, if*

1. $\mathcal{E} \subseteq \mathcal{E}_u^{\Pi}(M)$,
2. $\mathcal{UR}_{\mathcal{T}}^{\Pi}(M) = \bigcup_{t \in \mathcal{E}} \mathcal{UR}_{\mathcal{T}}^{\Pi}(tr(M,t))$
3. *and there does not exist an untimed transition $\hat{t} \in \mathcal{E}$ such that*

$$\mathcal{UR}_{\mathcal{T}}^{\Pi}(M) = \bigcup_{t \in \mathcal{E} \setminus \{\hat{t}\}} \mathcal{UR}_{\mathcal{T}}^{\Pi}(tr(M,t))$$

Definition 11 *For an original policy space Π, a refined policy space $\hat{\Pi} \subseteq \Pi$ that maintains the performance potential of Π is of* Π-irreducible support*, if for any vanishing marking $M \in \mathcal{R}_{\mathcal{V}}^{\Pi}$, the set $\mathcal{E}_u^{\hat{\Pi}}(M)$ is Π-irreducible with respect to M.*

Algorithm 2 Computing the target set $\mathcal{E}_u^{\hat{\Pi}_2}(M)$ for a given vanishing marking M

Input: The GSPN structure, Π_2, vanishing marking M.
Output: The transition set $\mathcal{E}_u^{\hat{\Pi}_2}(M)$.

```
1: E ← E_u^{Π2}(M)
2: for i = 1 → |E| do
3:    for j = 1 → C(i, |E|) do
4:       E1 ← the j-th subset of E of cardinality i, where these subsets are enumerated
         according to some rules that order the subsets of E with the same cardinality.
5:       if ⋃_{t∈E1} UR_T^{Π2}(tr(M,t)) = UR_T^{Π2}(M) then
6:          return E1
7:       end if
8:    end for
9: end for
```

Algorithm 2 provides a general procedure for the identification of a Π_2-irreducible set $\mathcal{E}_u^{\hat{\Pi}_2}(M)$ at any given marking M. Essentially, Algorithm 2 enumerates the subsets of the corresponding set $\mathcal{E}_u^{\Pi_2}(M)$ in increasing cardinality, and for each constructed subset, it checks the satisfaction of the condition of Equation (25). Subsets of equal cardinality are enumerated according to some pre-specified order; a lexicographic ordering based on the natural numbering of the elements of $\mathcal{E}_u^{\Pi_2}(M)$ can be used in lack of any other more pertinent order. The algorithm terminates when it encounters the first subset that satisfies the condition of Equation (25).

To alleviate the employed notation, and in line with the notation that is adopted in the statement of Algorithm 2, in the subsequent discussions, we shall use the notation $\mathcal{E}_1$ as an "alias" of $\mathcal{E}_u^{\hat{\Pi}_2}(M)$; and accordingly, we shall also define $\mathcal{E}_2 \equiv \mathcal{E}_u^{\Pi_2}(M) \setminus \mathcal{E}_1$.

Complexity considerations for the general refinement algorithm Algorithm 2 includes the enumeration of the Π_2-untimed reaches, which is a task with super-polynomial worst-case complexity with respect to the RAS size $|\Phi|$. Furthermore, the algorithm involves an enumeration of the subsets of $\mathcal{E}_u^{\Pi_2}(M)$, which are exponentially many with respect to $|\mathcal{E}_u^{\Pi_2}(M)|$.

However, when viewed from a more practical standpoint, the enumeration of the

Π_2-untimed reach of any given marking M is a local computation in the context of the underlying STD. These sub-graphs encode the potential evolution of the marking of the RAS-modeling GSPN in response to the completion of some processing task. This evolution involves the potential advancement of the process corresponding to the completed task, and possibly the advancement and/or the initiation of some additional processes that were blocked in the previous RAS operational state. The extent of this blocking depends on the structure of the underlying RAS, but in most practical cases, it will be limited to localized dependencies that are defined by the underlying RAS structure and the logic of the applied supervisory control policy. For all these reasons, the enumeration of the Π_2-untimed reaches by Algorithm 2 is expected to be a pretty tractable task, a fact that is confirmed by our computational experience.

Furthermore, these Π_2-untimed tangible reaches need not be calculated every time that Algorithm 2 is called. More specifically, in the implementation of the simulation-based optimization methods to be considered in Chapter 5, once a timed transition t_t is fired at a tangible marking $M_{\mathcal{T}}$ and a marking M is reached by this firing, the computer program can record the Π_2-untimed tangible reaches for all the markings in the Π_2-untimed vanishing reach of M, i.e., it can compute the set $\{\mathcal{UR}_{\mathcal{T}}^{\Pi_2}(M_{\mathcal{V}}) : M_{\mathcal{V}} \in \mathcal{UR}_{\mathcal{V}}^{\Pi_2}(M)\}$, and then employ this information at every invocation of Algorithm 2 as the system advances to the next tangible marking. Once the next tangible marking is reached, a new computational phase will begin; this computational phase will correspond to a new "macro-transition" of the underlying CTMC $\mathcal{M}$, and, thus, the previously constructed set of the Π_2-untimed tangible reaches will be discarded.

As for the enumeration of the subsets of the set $\mathcal{E}_u^{\Pi_2}(M)$, the reader should notice that it is only partial, since it is terminated as soon as Algorithm 2 identifies a set that satisfies the condition of Equation (25). Also, as it will be explained in the next example, the minimal cardinality of these successful sets is strongly dependent

upon the extent of the "conflict" that exists among the transitions of $\mathcal{E}_u^{\Pi_2}(M)$. If it is possible to identify a small set of conflicting transitions that are, however, in a non-conflicting relation with the rest of the transitions in $\mathcal{E}_u^{\Pi_2}(M)$, then, Algorithm 2 will not need to enumerate a large number of subsets of $\mathcal{E}_u^{\Pi_2}(M)$; and the computational experiments that are reported in later parts of this chapter indicate that this is actually the case for most vanishing markings M.

Example 3 *Applying Algorithm 2 at a specific vanishing marking of the GSPN of Example 1.*

A first role of this example is to illustrate the implementation of Algorithm 2. Furthermore, the provided elaborations on this example will also reveal a connection of the refining process that is pursued in this section and of the defining logic of Algorithm 2 with a notion of "(non-)conflict" that will be articulated in the subsequent discussion.

The GSPN structure and the particular marking considered in this example are those depicted in Figure 4.1. In fact, the reader might notice that the depicted GSPN and the applied policy space are the same as in Example 1 of Section 3.2. In particular, the marking presented in Figure 4.1 corresponds to vanishing marking #25 in Table B.1 of Appendix B, and it will be denoted by M_1 in the sequel.

The local STD that is defined by the Π_2-untimed reach of marking M_1 is depicted in Figure 4.2. This figure adopts the same semantics as Figure 3.6, i.e., the nodes depicted as single-bordered correspond to vanishing markings, and those depicted as double-bordered correspond to tangible markings. Furthermore, for each vanishing marking M in the Π_2-untimed reach, the set $\mathcal{UR}_{\mathcal{T}}^{\Pi_2}(M)$ is shown next to the node corresponding to M. In the implementation of Algorithm 2 in this example, subsets of the same cardinality are enumerated in the lexicographical order that is defined by the natural numbering of the transition set in the GSPN. Then, the nodes and arcs

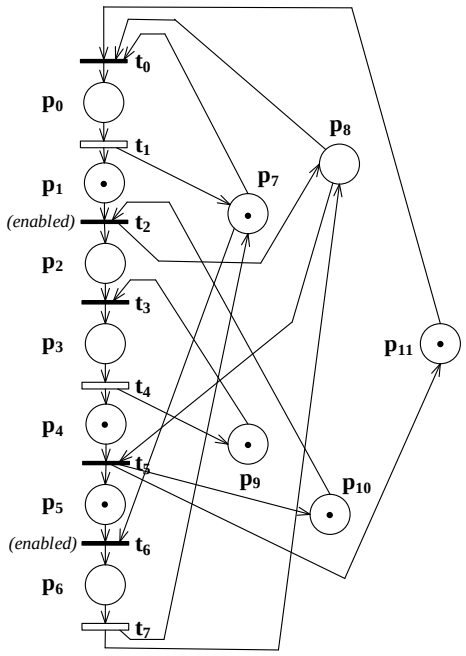


Figure 4.1: The marking M_1 of Example 3

that are boldfaced in Figure 4.2 represent the untimed reach of the considered vanishing marking M_1 under the refined policy space $\hat{\Pi}_2$ that applies the aforementioned implementation of Algorithm 2 for the refinement process.

From Figure 4.2 we can see that the refined policy space $\hat{\Pi}_2$ employs only one non-trivial random switch for the transition of the underlying GSPN from marking M_1 to its untimed tangible reach; this is the random switch at the marking M_3, where the set $\mathcal{E}_u^{\hat{\Pi}_2}(M_3)$ contains the enabled untimed transitions t_0 and t_5. These two transitions lead respectively to the tangible markings M_{T1} and M_{T2} that constitute the untimed tangible reach of M_1.

It is evident from the above discussion that the random switch refinement that is effected by Algorithm 2 at the various vanishing markings where this algorithm is applied, can be quite dramatic. Next, we take a closer look at the refinement process that is depicted in Figure 4.2, in an effort to reveal the detailed computation that is effected by Algorithm 2 at the various visited markings in Figure 4.2. But this

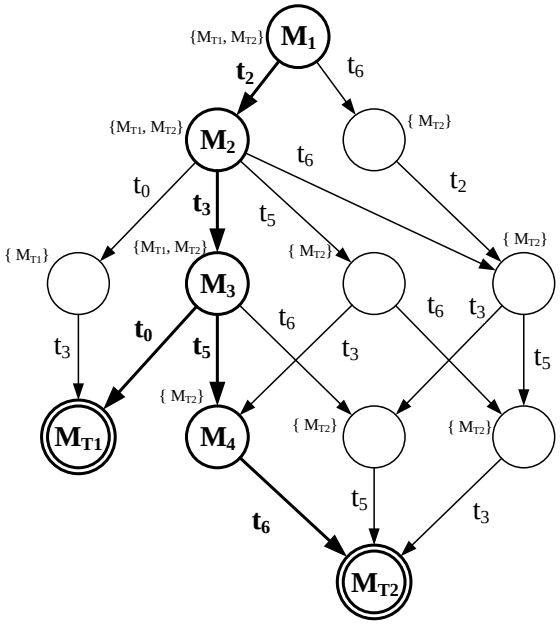


Figure 4.2: The local state transition diagram of $\mathcal{UR}^{\Pi_2}(M_1)$ in Example 3 and the sets of untimed tangible reaches of the vanishing nodes

discussion will also define a notion of "transition (non-)conflict" that can interpret the various choices that are made by the algorithm.

Hence, starting at marking M_1, we can see that $\mathcal{E}_u^{\Pi_2}(M_1) = \{t_2, t_6\}$, and the first subset of $\{t_2, t_6\}$ in the enumeration of increasing cardinality and lexicographical order is the singleton $\{t_2\}$. The set $\{t_2\}$ passes the test at Line 5 of Algorithm 2, since $\mathcal{UR}_{\mathcal{T}}^{\Pi_2}(tr(M_1, t_2)) = \{M_{T1}, M_{T2}\} = \mathcal{UR}_{\mathcal{T}}^{\Pi_2}(M_1)$. Therefore, the algorithm outputs the refined set $\mathcal{E}_u^{\hat{\Pi}_2}(M_1) = \{t_2\}$. At this point, the reader should also notice that the transitions t_2 and t_6 are not in conflict, in the classical sense of the DES / PN theory [94], since $\bullet t_2 = \{p_1, p_{10}\}$, and $\bullet t_6 = \{p_5, p_7\}$. However, in the considered context, the selection between these two non-conflicting transitions is still quite subtle. More specifically, it can be observed that in the GSPN structure of Figure 4.1, the firing of transition t_2 enables transition t_0 by releasing a token into place p_8. Meanwhile, t_0 shares place p_7 as a common input place with transition t_6. And this shared input

place with only one token implies a conflict between the transitions t_0 and t_6. Hence, if transition t_6 is given arbitrary precedence over transition t_2 at marking M_1, then the only token in place p_7 is consumed and transition t_0 cannot become enabled before the considered GSPN reaches a tangible marking. As a result, the untimed tangible reach of the marking $tr(M_1, t_6)$ loses the tangible marking M_{T1} that requires transition t_0 to be contained in the firing sequence from M_1 to M_{T1}. On the other hand, the firing of transition t_6 only releases one token to place p_6, which is not an input place of any untimed transitions. Hence, there is no potential conflict of the nature that was discussed above between any other untimed transition and transition t_2, and the dropping of transition t_6 from the refined set $\mathcal{E}_u^{\hat{\Pi}_2}(M_1)$ keeps unaltered the untimed tangible reach of M_1.

The second invocation of Algorithm 2 takes place at marking M_2 that results from firing transition t_2 at marking M_1. The entire set $\mathcal{E}_u^{\Pi_2}(M_2)$ is equal to $\{t_0, t_3, t_5, t_6\}$, and the refined set $\mathcal{E}_u^{\hat{\Pi}_2}(M_2)$ is set equal to $\{t_3\}$ in two steps: (i) the first candidate set, $\{t_0\}$, fails the test of Equation (25), since $\mathcal{UR}_{\mathcal{T}}^{\Pi_2}(tr(M_2, t_0)) = \{M_{T1}\} \neq \mathcal{UR}_{\mathcal{T}}^{\Pi_2}(M_2)$; (ii) on the other hand, the second candidate set, $\{t_3\}$, passes the test. From the perspective of the potential conflicts that was discussed in the previous paragraph, first we notice that none of the transitions in $\mathcal{E}_u^{\Pi_2}(M_2) = \{t_0, t_3, t_5, t_6\}$ can enable new untimed transitions upon its firing, according to the Π_2-conditional STD depicted in Figure 4.2. Next, it can also be seen that the two subsets $\{t_3\}$ and $\{t_0, t_5, t_6\}$ are not in conflict, since $\bullet t_3 = \{p_2, p_9\}$, and $\bullet t_0 \cup \bullet t_5 \cup \bullet t_6 = \{p_4, p_5, p_7, p_8, p_{11}\}$. Hence, in this case, either of the sets $\{t_3\}$ and $\{t_0, t_5, t_6\}$ can be selected as the refined set $\mathcal{E}_u^{\hat{\Pi}_2}(M_2)$ and the untimed tangible reach will not change. Algorithm 2 picks the set $\{t_3\}$ since it has the smaller cardinality.

Finally, at marking M_3, all the singleton subsets fail the test, but $\{t_0, t_5\}$, the first subset of cardinality two, passes the test and it is set as $\mathcal{E}_u^{\hat{\Pi}_2}(M_3)$. Under this refined random switch, from marking M_3 we can reach either the tangible marking M_{T1} by

firing transition t_0, or the vanishing marking M_4 (and finally the tangible marking M_{T2}) by firing transition t_5. From the perspective of the potential conflicts that were discussed in the previous paragraphs, we can see that transition t_0 is in conflict with transitions t_5 and t_6, but t_5 and t_6 are not in conflict. Therefore, only one of the transitions t_5 and t_6 should be kept in the refined set $\mathcal{E}_u^{\hat{\Pi}_2}(M_3)$. And Algorithm 2 removes transition t_6 from $\mathcal{E}_u^{\hat{\Pi}_2}(M_3)$ due to the application of the lexicographical order.

The next section will elaborate further on the connection that was illustrated by this example, between the notion of "(non-)conflict" and the refinement criterion employed at Line 5 of Algorithm 2.

4.1.3 Some sufficient conditions on $\mathcal{E}_u^{\hat{\Pi}_2}(M)$ that maintain the corresponding untimed tangible reach

The main theme of this section is the development of some sufficient conditions on $\mathcal{E}_u^{\hat{\Pi}_2}(M)$ that can guarantee the satisfaction of the condition of Equation (25) while foregoing the complete enumeration of the corresponding untimed reach. Ideally, we would like these conditions to have a *structural* character, i.e., they should be able to infer the ability of a candidate set $\mathcal{E}_1$ to maintain the untimed tangible reach of the corresponding vanishing marking M based only on the structural information of the underlying GSPN and the marking M itself, instead of employing a process that involves the enumeration of all the possible firing sequences of untimed transitions emanating from marking M. The conditions that are derived in this section do not possess exactly the structural flavor that was described above, but, as we shall see in some subsequent parts of this chapter, they provide a foundation for such an effective structural analysis when adapted in the context of some specially structured RAS, like the CRL that is considered in this document.

Furthermore, the subsequent developments are based on the notion of "(non-)conflict" that was introduced in the discussion of Example 3. In particular, at any

vanishing marking M, we are seeking a partition of the transition set $\mathcal{E}_u^{\Pi_2}(M)$ into two subsets $\mathcal{E}_1$ and $\mathcal{E}_2$, such that transitions in $\mathcal{E}_1$ are not in conflict either with the transitions in $\mathcal{E}_2$, or with any untimed transitions that become enabled by firing some feasible transition sequences composed by untimed transitions not in $\mathcal{E}_1$. The following proposition, that first appeared in [60], provides a formal characterization for the notion of "conflict" that is pertinent in the context of our application, and reveals its connection to Equation (25).

To formally state and prove this proposition, we also need a new operator $\overline{\mathcal{E}_u^{\Pi}}(\cdot,\cdot)$. For a vanishing marking M, a set of untimed transitions $\tilde{T}$, and a policy space Π, the operator $\overline{\mathcal{E}_u^{\Pi}}(M,\tilde{T})$ denotes the set containing the untimed transitions that are not Π-enabled at M but get enabled after firing some Π-feasible untimed transition sequences in $\tilde{T}^*$; i.e., $\forall t \in \overline{\mathcal{E}_u^{\Pi}}(M,\tilde{T})$: (i) $t \notin \mathcal{E}_u^{\Pi}(M)$; and (ii) $\exists \sigma \in \tilde{T}^*$ such that σ is Π-feasible and $t \in \mathcal{E}_u^{\Pi}(tr(M,\sigma))$.

Proposition 5 *[60] Consider a vanishing marking M and a partition $(\mathcal{E}_1,\mathcal{E}_2)$ of $\mathcal{E}_u^{\Pi_2}(M)$, and let $\hat{\mathcal{E}}_2 \equiv \mathcal{E}_2 \cup \overline{\mathcal{E}_u^{\Pi_2}}(M,\mathcal{T}_u \setminus \mathcal{E}_1)$. Suppose that for each Π_2-feasible sequence $\sigma \in \hat{\mathcal{E}}_2^*$, there exists a transition sequence $\hat{\sigma} \in \hat{\mathcal{E}}_2^*$ and a transition $\hat{t} \in \mathcal{E}_1$, such that both sequences $\sigma\hat{\sigma}\hat{t}$ and $\hat{t}\sigma\hat{\sigma}$ are Π_2-feasible at M. Then, setting $\mathcal{E}_1$ as $\mathcal{E}_u^{\hat{\Pi}_2}(M)$ can satisfy the condition of Equation* (25).

Proof: We shall show that $\mathcal{UR}_{\mathcal{T}}^{\Pi_2}(M) = \bigcup_{t\in\mathcal{E}_1} \mathcal{UR}_{\mathcal{T}}^{\Pi_2}(tr(M,t))$. The inclusion $\mathcal{UR}_{\mathcal{T}}^{\Pi_2}(M) \supseteq \bigcup_{t\in\mathcal{E}_1} \mathcal{UR}_{\mathcal{T}}^{\Pi_2}(tr(M,t))$ is obvious, since $\mathcal{E}_1 \subseteq \mathcal{E}_u^{\Pi_2}(M)$. So, next we show that $\mathcal{UR}_{\mathcal{T}}^{\Pi_2}(M) \subseteq \bigcup_{t\in\mathcal{E}_1} \mathcal{UR}_{\mathcal{T}}^{\Pi_2}(tr(M,t))$.

Consider any tangible marking $M_{\mathcal{T}} \in \mathcal{UR}_{\mathcal{T}}^{\Pi_2}(M)$. Then, there exists a Π_2-feasible untimed transition sequence $\sigma^n = t_1 t_2 \dots t_n$ such that $M \xrightarrow{\sigma} M_{\mathcal{T}}$. To show that $M_{\mathcal{T}} \in \bigcup_{t\in\mathcal{E}_1} \mathcal{UR}_{\mathcal{T}}^{\Pi_2}(tr(M,t))$, we distinguish the following two cases:

Case 1: $t_1 \in \mathcal{E}_1$. Then, $M_{\mathcal{T}} \in \mathcal{UR}_{\mathcal{T}}^{\Pi_2}(tr(M,t_1)) \subseteq \bigcup_{t\in\mathcal{E}_1} \mathcal{UR}_{\mathcal{T}}^{\Pi_2}(tr(M,t))$.

Case 2: $t_1 \in \mathcal{E}_2$. Then, by the last assumption of Proposition 5, there exists $t_i \in \mathcal{E}_1$ in the sequence σ^n such that the transition sequence $\sigma^{i-1} = t_1 \dots t_{i-1} \in \hat{\mathcal{E}}_2^*$.

According to the same assumption, the firing order of t_i and σ^{i-1} can be exchanged and get the transition sequence $t_i t_1 \dots t_{i-1}$ which is Π_2-feasible at M.

But then, the sequence $t_i t_1 \dots t_{i-1} t_{i+1} \dots t_n$ is a Π_2-feasible sequence leading from marking M to $M_{\mathcal{T}}$, so $M_{\mathcal{T}} \in \mathcal{UR}_{\mathcal{T}}^{\Pi_2}(tr(m, t_i)) \subseteq \bigcup_{t \in \mathcal{E}_1} \mathcal{UR}_{\mathcal{T}}^{\Pi_2}(tr(m, t))$. $\square$

The result of Proposition 5 is not a fully structural condition since it necessitates an enumeration of the Π_2-feasible untimed transition sequences in $\hat{\mathcal{E}}_2^*$. However, some special cases where this condition might be easily tested, is when it is possible to guarantee that the firing of any transition in $\mathcal{E}_2$ will not enable any new untimed transitions in $\mathcal{T}_u \setminus \mathcal{E}_u^{\Pi_2}(M)$. The next proposition establishes such a special case.

Proposition 6 *Consider a vanishing marking M and a partition $(\mathcal{E}_1, \mathcal{E}_2)$ of $\mathcal{E}_u^{\Pi_2}(M)$, and further suppose that:*

(i). $\exists \hat{t} \in \mathcal{E}_1$ *such that* $\forall t \in \mathcal{E}_2,\ {}^\bullet\hat{t} \cap {}^\bullet t = \emptyset$.

(ii). $\forall t \in \mathcal{E}_2$ *and* $\forall p \in \mathcal{P},\ \ p \in t^\bullet \implies p^\bullet \cap \mathcal{T}_u = \emptyset$.

(iii). The constraints imposed by the policy space Π_2 do not impact the fireability of any transitions in $\{\hat{t}\} \cup \mathcal{E}_2$ at any vanishing marking reachable by some Π_2-feasible sequence of $\{\hat{t}\} \cup \mathcal{E}_2$; i.e., let $\mathcal{E}_2' = \{\hat{t}\} \cup \mathcal{E}_2$, then for any Π_2-feasible sequence $\sigma \in \mathcal{E}_2'^*$ at M, and any $t \in \mathcal{E}_2'$, $t \in \mathcal{E}_u^{\Pi_0}(tr(M, \sigma)) \Rightarrow t \in \mathcal{E}_u^{\Pi_2}(tr(M, \sigma))$.[3]

Then, setting $\mathcal{E}_1$ as $\mathcal{E}_u^{\hat{\Pi}_2}(M)$ can satisfy the condition of Equation (25).

Proof: We shall prove the result of Proposition 6 by showing that the three conditions stated in this proposition are sufficient to satisfy the requirements of Proposition 5.

[3]We remind the reader that an important set of the constraints defining policy space Π_2 pertains to the liveness-enforcement of the underlying RAS-modeling GSPN. We also notice that condition (iii) of this proposition will be redundant in the case that the applied supervisory control policy is encoded through the structure of the RAS-modeling GSPN along the lines that were discussed in Section 3.1.1.

We start by noticing that condition (ii) guarantees that the firing of any sequence $\sigma \in \mathcal{E}_2^*$ does not add any tokens to the input places of any untimed transition. Therefore, under this condition, $\hat{\mathcal{E}}_2 = \mathcal{E}_2$, where the set $\hat{\mathcal{E}}_2$ is defined in the statement of Proposition 5, and we only need to show that $\forall \sigma \in \mathcal{E}_2^*$, there exist $\hat{\sigma} \in \mathcal{E}_2^*$ and $\hat{t} \in \mathcal{E}_1$ such that both sequences $\sigma\hat{\sigma}\hat{t}$ and $\hat{t}\sigma\hat{\sigma}$ are Π_2-feasible at M.

Consider any sequence $\sigma = t_1 t_2 \dots t_n \in \mathcal{E}_2^*$ that is Π_2-feasible at M. From condition (i), $\bullet\hat{t} \cap \bullet t_i = \emptyset,\ \forall i \in \{1, 2, \dots, n\}$. Therefore, firing σ does not decrease the number of tokens in any input place of $\hat{t}$. This last remark, when combined with condition (iii) of the considered proposition, implies that $\hat{t}$ is feasible in the marking $M' = tr(M, \sigma)$, or, in other words, that the transition sequence $\sigma\hat{t}$ is Π_2-feasible at M. The feasibility of $\hat{t}\sigma$ can be proved in a similar manner. Therefore, the condition of Proposition 5 is met by setting $\hat{\sigma} = \varepsilon$. $\square$

In the next section, we employ Proposition 6 in order to define a very efficient and pertinent policy space for the GSPN that models the CRL operations.

4.1.4 Specialization to the CRL case

As discussed in the previous section, in general, it is difficult to find a generic refinement algorithm that can maintain the performance potential of the original policy space while avoiding the enumeration of the firing sequences of untimed transitions at each visited vanishing marking. However, for the specialized setting of the CRL, the requested refinement can be performed through some structural analysis.

We start the presentation of the corresponding developments by reminding the reader that in the CRL-modeling GSPN that was described in Section 3.2, the untimed transitions can be classified into three types:

Type I – buffer transfer: This subset of $\mathcal{T}_u$ consists of the untimed transitions

modeling the advancement of jobs that have completed processing at their current workstation to their next workstation. These advancements must be enabled by the applied deadlock avoidance constraints, and involve the allocation of a buffer slot in the requested workstation and the eventual release of the held buffer slot at the current workstation. For further reference, the set of all Type I transitions will be denoted by $\mathcal{T}_I$.

Type II – process start: This subset of $\mathcal{T}_u$ consists of the untimed transitions modeling the server allocation to a job waiting for processing at one of the line stations. The reader should notice that no resources are released upon the firing of these transitions. The set of all Type II transitions will be denoted by $\mathcal{T}_{II}$.

Type III – load: This subset of $\mathcal{T}_u$ is the singleton $\mathcal{T}_{III}$ containing only the untimed transition that models the loading of a new job into the line and the allocation to this job of one buffer slot and the server of workstation $\mathcal{W}(1)$; this transition will be denoted as t_0 in the sequel. Furthermore, the corresponding event must be admissible by the applied DAP. Finally, the reader should also notice that the firing of the transition t_0 does not involve any release of previously allocated resources.

If a transition t is Type II or Type III, then it satisfies condition (ii) of Proposition 6 and it may become a candidate for the subset $\mathcal{E}_2$, which collects the enabled untimed transitions that can be treated as "redundant options" in the corresponding vanishing marking. A simpler case is that where all the enabled untimed transitions of the current marking M are Type II or Type III:

Proposition 7 *For a vanishing marking M, if $\mathcal{E}_u^{\Pi_2}(M) \cap \mathcal{T}_I = \emptyset$, and the set $\mathcal{E}_u^{\hat{\Pi}_2}(M)$*

collects all the transitions corresponding to the server allocation of a single workstation,[4] *then*

(i). $\mathcal{E}_u^{\hat{\Pi}_2}(M)$ *satisfies the condition of Equation* (25)*, and*

(ii). $\mathcal{E}_u^{\hat{\Pi}_2}(M)$ *is a* Π_2*-irreducibly refined set with respect to* M*.*

In order to prove the result of Proposition 7, we need the following lemma:

Lemma 1 *Consider a marking* M *of a CRL-modeling GSPN* $\mathcal{N}$*, and suppose that there exist two untimed transition sequences,* $\sigma_1, \sigma_2 \in \mathcal{T}_u^*$*, that are both* Π_2*-feasible at* M *and result at the same marking* M'*. Then,* $\vec{\sigma}_1 = \vec{\sigma}_2$*, where* $\vec{\sigma}_i$*,* $i = 1, 2$*, denotes the Parikh vector of the corresponding sequence* σ_i*.*

Proof: According to the remark at the end of the opening part of Section 3.1, in a RAS-modeling GSPN any reachable marking M is uniquely defined by the submarking $\hat{M}$ of the process places of the net.

Furthermore, in a CRL-modeling GSPN, each transition $t \in \mathcal{T}_u$ can increase (reps., decrease) the token content of at most one place in the aforementioned place subset, and these places are distinct for different transitions. But then, the validity of Lemma 1 results from the additional fact that, in the considered GSPN class, any Π_2-feasible sequence in $\mathcal{T}_u^*$ does not involve any cyclic behavior (c.f. Proposition 3 in Section 3.1.2). □

Next, we proceed with the proof of Proposition 7.

Proof of Proposition 7: Let $\mathcal{E}_1 \equiv \mathcal{E}_u^{\hat{\Pi}_2}(M)$ as defined in Proposition 7, and $\mathcal{E}_2 \equiv \mathcal{E}_u^{\Pi_2}(M) \setminus \mathcal{E}_1$. Without loss of generality, suppose $\mathcal{E}_2 \neq \emptyset$.

To prove the first conclusion in Proposition 7, we can verify the three conditions of Proposition 6.

[4]We remind the reader that the only transitions that involve buffer allocation are the Type I transitions and transition t_0. Therefore, given the absence of Type I transitions in the set $\mathcal{E}_u^{\Pi_2}(M)$, transition t_0 can be in conflict with other transitions only through the server allocation.

The transitions in $\mathcal{E}_2$ correspond to server allocation in other workstations than the transitions in $\mathcal{E}_1$, and thus, they do not have any common input places with the transitions in $\mathcal{E}_1$; so, condition (i) is satisfied. Condition (ii) is satisfied since, under the working assumptions, (a) the tokens that are advanced by the firing of some transition in $\mathcal{E}_2$ are moving to a process place p with only one output transition which is timed, and furthermore, (b) no transitions in $\mathcal{E}_u^{\Pi_2}(M)$ can release any resources.

For the following needs of this proof, we also remark / emphasize the following: The previous discussion established that the firing of the transitions in $\mathcal{E}_2$ cannot enable new untimed transitions. Furthermore, the single-server assumption for the considered CRL configurations also implies that the fired transitions cannot be enabled again until a tangible marking is reached. When taken together, the above remarks imply that $\forall M' \in \mathcal{UR}_{\mathcal{V}}^{\Pi_2}(M) \setminus \{M\}$, $\mathcal{E}_u^{\Pi_2}(M')$ is a proper subset of $\mathcal{E}_u^{\Pi_2}(M)$.

To verify condition (iii) of Proposition 6, first note that the deadlock avoidance policy cannot affect the enabling of Type II transitions. Next we consider the verification of condition (iii) for the case that $t_0 \in \mathcal{E}_u^{\Pi_2}(M)$, where t_0 is the unique element in $\mathcal{T}_{III}$. In this case, the deadlock avoidance policy permits the loading of a new job instance into the CRL at marking M. For any given $M' \in \mathcal{UR}_{\mathcal{V}}^{\Pi_2}(M)$ and $t_0 \in \mathcal{E}_u^{\Pi_0}(M')$, we shall prove that $t_0 \in \mathcal{E}_u^{\Pi_2}(M')$.

Suppose that σ is a Π_2-feasible sequence from M to M'. Since $t_0 \in \mathcal{E}_u^{\Pi_2}(M')$ and t_0 allocates the single server of workstation $\mathcal{W}(1)$, t_0 cannot be in sequence σ. And from the fact that $\forall M'' \in \mathcal{UR}_{\mathcal{V}}^{\Pi_2}(M)$, $\mathcal{E}_u^{\Pi_2}(M'') \subseteq \mathcal{E}_u^{\Pi_2}(M)$, σ contains only Type II transitions. Therefore, the simplified RAS states $\hat{s}$ corresponding to M and M' are the same. But then, the deadlock avoidance policy also permits the firing of t_0 at marking M', i.e., $t_0 \in \mathcal{E}_u^{\Pi_2}(M')$, and condition (iii) is verified.

For the second conclusion of Proposition 7, since all transitions in $\mathcal{E}_u^{\hat{\Pi}_2}(M)$ correspond to the server allocation at one workstation, the firing of any transition in this set will allocate the corresponding server and will disable all the other transitions.

Furthermore, as established in the earlier parts of this proof, the firing of any transitions in $\mathcal{E}_u^{\Pi_2}(M)$ cannot enable new untimed transitions. Therefore, the disabled untimed transitions in $\mathcal{E}_u^{\hat{\Pi}_2}(M)$ cannot become enabled before reaching a new tangible marking. In other words, if an untimed transition of $\mathcal{E}_u^{\hat{\Pi}_2}(M)$ is not fired at M, then it cannot be fired later in any sequence of untimed transitions that leads to a tangible marking. This conclusion, when combined with Lemma 1, establishes the sought result. □

Next we consider the case where Type I transitions are also enabled at the current marking. Intuitively, since conflicts cannot happen between Type I and Type II transitions, and the firing of any Type II transitions cannot enable new untimed transitions, it is always "safe" to eliminate Type II transitions when $\mathcal{E}_u^{\Pi_2}(M)$ has only Type I and Type II transitions. In the case that $t_0 \in \mathcal{E}_u^{\Pi_2}(M)$ and it is not in conflict with any Type I transition, then t_0 can be eliminated as well. Otherwise, t_0 should be kept in $\mathcal{E}_u^{\hat{\Pi}_2}(M)$.

The next condition will help us formalize the above remarks.

Condition 4 *For a given vanishing marking M, $t_0 \in \mathcal{E}_u^{\Pi_2}(M)$, and there exists a transition $\hat{t} \in \mathcal{E}_u^{\Pi_2}(M) \cap \mathcal{T}_I$, such that $\hat{t} \in \mathcal{E}_u^{\Pi_2}(tr(M, t_0))$ and $t_0 \in \mathcal{E}_u^{\Pi_2}(tr(M, \hat{t}))$.*

Condition 4 enables the following formal statement of the more intuitive suggestions that were provided above regarding the structuring of the set $\mathcal{E}_u^{\hat{\Pi}_2}(M)$ in the case that the corresponding set $\mathcal{E}_u^{\Pi_2}(M)$ contains type I transitions.

Proposition 8 *For a vanishing marking M, if $\mathcal{E}_u^{\Pi_2}(M) \cap \mathcal{T}_I \neq \emptyset$, and*

$$\mathcal{E}_u^{\hat{\Pi}_2}(M) = \begin{cases} \mathcal{E}_u^{\Pi_2}(M) \cap \mathcal{T}_I & \text{if } t_0 \notin \mathcal{E}_u^{\Pi_2}(M) \text{ or Condition 4 holds} \\ (\mathcal{E}_u^{\Pi_2}(M) \cap \mathcal{T}_I) \cup \{t_0\} & \text{otherwise} \end{cases} \tag{26}$$

then $\mathcal{E}_u^{\hat{\Pi}_2}(M)$ satisfies the condition of Equation (25).

Proof: If Condition 4 holds, then this proposition can be proved by verifying the condition of Proposition 5. More specifically, let $\mathcal{E}_2 \equiv \mathcal{E}_u^{\Pi_2}(M) \setminus \mathcal{E}_u^{\hat{\Pi}_2}(M)$ and $\hat{\mathcal{E}}_2 \equiv \mathcal{E}_2 \cup \overline{\mathcal{E}_u^{\Pi_2}}(M, \mathcal{T}_u \setminus \mathcal{E}_u^{\hat{\Pi}_2}(M))$. Then, we shall prove that for any Π_2-feasible sequence $\sigma \in \hat{\mathcal{E}}_2^*$, there exists a transition $\hat{t} \in \mathcal{E}_u^{\hat{\Pi}_2}(M)$ and a sequence $\hat{\sigma} \in \hat{\mathcal{E}}_2^*$ such that both $\sigma\hat{\sigma}\hat{t}$ and $\hat{t}\sigma\hat{\sigma}$ are Π_2-feasible at M. Furthermore, from Equation (26), the set $\mathcal{E}_2$ does not contain any Type I transitions. Hence, no new untimed transitions can be enabled by firing any transition sequence in $\mathcal{E}_2^*$ at M, and thus, we can set $\hat{\mathcal{E}}_2 = \mathcal{E}_2$.

Consider any transition sequence $\sigma = t_1 t_2 \ldots t_n \in \mathcal{E}_2^*$ that is Π_2-feasible at M. We will show that the transition $\hat{t}$ of Condition 4 is enabled at marking $M' \equiv tr(M, \sigma)$, or, in other words, the transition sequence $\sigma\hat{t}$ is Π_2-feasible at M. Indeed, if transition t_0 is not contained in the sequence σ, then transition $\hat{t}$ is enabled at marking M' since (i) $\hat{t}$ does not share any common input places with any Type II transitions, and (ii) the firing of any Type II transitions does not change the simplified RAS state corresponding to the GSPN marking. Even if transition t_0 is contained in the sequence σ, $\hat{t}$ is still enabled at marking M', because t_0 can appear only once in σ, and Condition 4 implies that the common input places of the transitions $\hat{t}$ and t_0 have enough tokens for firing both of them. Furthermore, Condition 4 also implies that the applied DAP permits the firing of $\hat{t}$ at the simplified RAS state corresponding to the marking $tr(M, t_0)$, and this marking does not change by firing the remaining transitions in the considered sequence σ. The Π_2-feasibility of the sequence $\hat{t}\sigma$ can be proved in a similar manner. Therefore, the condition of Proposition 5 is met by setting $\hat{\sigma} = \varepsilon$.

On the other hand, if Condition 4 does not hold, then the proposition can be proved by verifying the three conditions of Proposition 6. In this case, either $t_0 \notin \mathcal{E}_u^{\Pi_2}(M)$ or $t_0 \in \mathcal{E}_u^{\hat{\Pi}_2}(M)$. Condition (ii) can be verified in the same way as in the proof of Proposition 7. And condition (i) of Proposition 6 is satisfied since Type I and Type II transitions have no common input places. Finally, since the firing of any

Algorithm 3 Computing the target set $\mathcal{E}_u^{\hat{\Pi}_2}(M)$ at a vanishing marking M of the CRL-modeling GSPN

Input: The GSPN structure, Π, vanishing marking M.
Output: The transition set $\mathcal{E}_u^{\hat{\Pi}_2}(M)$.

1: $\mathcal{E} \leftarrow \mathcal{E}_u^{\Pi_2}(M)$
2: **if** $\mathcal{E} \cap \mathcal{T}_I = \emptyset$ **then**
3: **return** $\{t \in \mathcal{E} : t$ models the server allocation at the workstation corresponding to the first transition in $\mathcal{E}\}$.
4: **end if**
5: **if** $t_0 \notin \mathcal{E}$ **then**
6: **return** $\mathcal{E} \cap \mathcal{T}_I$.
7: **end if**
8: **for all** $t \in \mathcal{E} \cap \mathcal{T}_I$ **do**
9: **if** $t \in \mathcal{E}_u^{\Pi_2}(tr(M, t_0)) \wedge t_0 \in \mathcal{E}_u^{\Pi_2}(tr(M, t))$ **then**
10: **return** $\mathcal{E} \cap \mathcal{T}_I$.
11: **end if**
12: **end for**
13: **return** $\mathcal{E} \cap (\mathcal{T}_I \cup \{t_0\})$.

Type II transitions cannot change the simplified RAS states, the validity of condition (iii) is obvious. □

The refined set $\mathcal{E}_u^{\hat{\Pi}_2}(M)$ that is defined in Proposition 8 will successfully remove the redundancy that is due to the presence of non-conflicting options in the original set $\mathcal{E}_u^{\Pi_2}(M)$ in most cases. But it may fail to identify some redundancy that takes the form of a "pseudo"-conflict. Therefore, unlike Proposition 7, the result of Proposition 8 does not guarantee Π_2-irreducibility. An example demonstrating this lack of Π_2-irreducibility, and revealing the corresponding notion of "pseudo"-conflict mentioned above, is provided in the example of Appendix C.

The results of Propositions 7 and 8 can be integrated in an algorithm that provides an alternative to Algorithm 2 for the CRL case. The main advantage of this algorithm is that it is no longer necessary to compute and record the untimed tangible reaches for the specification of the refined random switches. Furthermore, since the rules that generate the CRL-modeling GSPN have been well-defined, Algorithm 3 can take either the original CRL or the corresponding GSPN as part of its input.

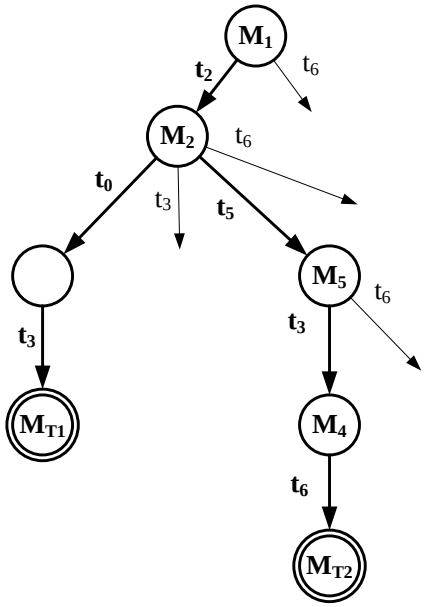


Figure 4.3: The local STD of $\mathcal{UR}^{\hat{\Pi}_2}(M_1)$ in Example 3 under Algorithm 3

Example 3 revisited In this example we go back to Example 3, where the marking M_1 in Figure 4.1 is a vanishing marking of the GSPN that models the CRL of Example 1. Figure 4.3 depicts the implementation of Algorithm 3 on marking M_1 and the $\hat{\Pi}_2$-untimed reach that is returned by this algorithm. The reader should notice that this time the Π_2-untimed reach is not enumerated, and the non-boldfaced arcs without destination nodes are the eliminated options of enabled transitions from the original policy space Π_2.

In the considered GSPN, there are five untimed transitions: t_0, t_2, t_3, t_5, t_6. According to the description in Table 3.1, $\mathcal{T}_I = \{t_2, t_5\}$, $\mathcal{T}_{II} = \{t_3, t_6\}$, $\mathcal{T}_{III} = \{t_0\}$. At marking M_1, $\mathcal{E}_u^{\Pi_2}(M_1) = \{t_2, t_6\}$; hence, t_6 is removed from $\mathcal{E}_u^{\hat{\Pi}_2}(M_1)$, since $t_6 \in \mathcal{T}_{II}$ and $t_2 \in \mathcal{T}_I$. As a result, only $M_2 = tr(M_1, t_2)$ is reachable in the policy space $\hat{\Pi}_2$. At marking M_2, $\mathcal{E}_u^{\Pi_2}(M_2) = \{t_0, t_3, t_5, t_6\}$. Since $t_3, t_6 \in \mathcal{T}_{II}$ and $t_5 \in \mathcal{T}_I$, t_3 and t_6 are removed from $\mathcal{E}_u^{\hat{\Pi}_2}(M_1)$ and t_5 is kept. With respect to the inclusion of transition t_0, since $t_5 \notin \mathcal{E}_u^{\Pi_2}(tr(M_2, t_0))$, Condition 4 does not hold, and thus, t_0 is also kept in $\mathcal{E}_u^{\hat{\Pi}_2}(M_1)$. The firing of t_0 at M_2 leads to the tangible marking M_{T1} without any

further possible branching. On the other hand, the firing of t_5 at M_2 leads to a vanishing marking M_5, where $\mathcal{E}_u^{\Pi_2}(M_5) = \{t_3, t_6\}$. Both of the two enabled untimed transitions are Type II. Thus, Algorithm 3 picks the transitions that share the same workstation as the first transition in the set $\mathcal{E}_u^{\Pi_2}(M_5)$. In this case, the first transition is t_3 and it is associated with Workstation #2. Therefore, t_3 is kept and t_6 is removed from $\mathcal{E}_u^{\hat{\Pi}_2}(M_5)$, since t_6 is associated with Workstation #1.

The untimed reach of M_1 obtained from Algorithm 3 is different from the one obtained from Algorithm 2 but it is equally good: there remains only one random switch of two options, corresponding to the two markings in the corresponding untimed tangible reach $\mathcal{UR}_{\mathcal{T}}^{\Pi_2}(M_1)$.

Even though the equivalence highlighted in the previous paragraph is not always true (c.f. the example in Appendix C), Algorithm 3 is a competitive algorithm to Algorithm 2 since its lack of Π_2-irreducibility with respect to certain vanishing markings does not impact substantially the effectiveness of the corresponding refinement process; this assessment is clearly corroborated by the results of an empirical study that is presented in the next section. Furthermore, since Algorithm 3 always maintains the untimed tangible reach of the processed marking and it executes fast due to the structural nature of the information that it employs about this marking, one can also envision the application of this algorithm as a pre-processing stage that will remove a significant part of the originally enabled untimed transitions before Algorithm 2 is eventually applied, ensuring thus the Π_2-irreducibility of the final result. Finally, in a later section of this chapter we shall also see that Algorithm 3 can even out-perform Algorithm 2 when they are both applied in a new policy space that is obtained from the further restriction of the policy space Π_2.

4.1.5 A numerical experiment for the random-switch refinement processes developed in this chapter

In this section we assess and validate the effectiveness of the algorithms that were developed in this chapter through a numerical experiment that employs the 20 CRL configurations in Table 4.1. These configurations were also used as a testbed for the developments in [60]. In Table 4.1, "WS" refers to "workstation(s)" and "JS" refers to "job stage(s)". The sizes of the included configurations span from the size of the small CRL in Example 1, to a moderate size of 5 workstations supporting 8 processing stages. These sizes have been selected so that the underlying state spaces remain fairly easily enumerable, for comparison purposes. We also note that Table 4.1 does not specify the processing rates of the various stages of the enlisted configurations, since the sizes of the state spaces and the numbers of the primary decision variables for the corresponding MP formulations are not dependent on these rates.[5]

The results of the applications of the two refinement algorithms on all the 20 CRL configurations are reported in Table 4.2. The column entitled " Tangible Markings" in this table reports the numbers of tangible markings for each CRL configuration, as a measure of the size of the corresponding state space, since the refinement process does not actually change the set of tangible markings. On the other hand, the last four columns in Table 4.2 show the results of the refinement process where the policy space Π_2 is replaced by the corresponding refined policy space $\hat{\Pi}_2$, in the definitions of the decision variables $\bar{\zeta}$, the index-searching function $idx(\cdot)$, and the objective function of the MP formulation (14)–(16).

Both of the refinement Algorithms 2 and 3 incur considerable reduction of the employed random switches and decision variables. Since Proposition 8 does not support the Π_2-irreducibility requirement, Algorithm 2 performs better than Algorithm

[5]However, if Assumption 4 is not true, then the sizes of the corresponding state spaces will be affected by the employed timing distributions due to the Markovian approximation of the non-Markovian timing distributions.

Table 4.1: The CRL configurations employed in the numerical experiment that is reported in Sections 4.1.5 and 4.2.2 [60].

Conf.	WS	job stages (JS) and job routes	workstation buffering capacities
1	2	3JS ($WS_1 \to WS_2 \to WS_1$)	$(B_1, B_2) = (2, 2)$
2	2	3JS ($WS_1 \to WS_2 \to WS_1$)	$(B_1, B_2) = (1, 2)$
3			$(B_1, B_2) = (3, 2)$
4			$(B_1, B_2) = (4, 4)$
5			$(B_1, B_2) = (10, 10)$
6	3	4JS ($WS_1 \to WS_2 \to WS_3 \to WS_1$)	$(B_1, B_2, B_3) = (1, 2, 2)$
7			$(B_1, B_2, B_3) = (3, 2, 2)$
8			$(B_1, B_2, B_3) = (4, 3, 2)$
9			$(B_1, B_2, B_3) = (5, 5, 6)$
10	4	7JS ($WS_1 \to WS_2 \to WS_4 \to WS_1 \to WS_2 \to WS_3 \to WS_1$)	$(B_1, B_2, B_3, B_4) = (3, 2, 1, 2)$
11	3	5JS ($WS_1 \to WS_2 \to WS_3 \to WS_1 \to WS_2$)	$(B_1, B_2, B_3) = (3, 4, 3)$
12	3	5JS ($WS_1 \to WS_2 \to WS_3 \to WS_2 \to WS_3$)	$(B_1, B_2, B_3) = (3, 3, 3)$
13	3	5JS ($WS_1 \to WS_2 \to WS_1 \to WS_3 \to WS_2$)	$(B_1, B_2, B_3) = (3, 4, 1)$
14			$(B_1, B_2, B_3) = (2, 2, 2)$
15	3	6JS ($WS_1 \to WS_2 \to WS_3 \to WS_1 \to WS_2 \to WS_3$)	$(B_1, B_2, B_3) = (2, 3, 2)$
16			$(B_1, B_2, B_3) = (2, 2, 2)$
17	4	7JS ($WS_1 \to WS_2 \to WS_4 \to WS_1 \to WS_2 \to WS_3 \to WS_1$)	$B_i = 3, i = 1, 2, \ldots, 5$
18	5	7JS ($WS_1 \to WS_2 \to WS_1 \to WS_3 \to WS_4 \to WS_5 \to WS_4$)	$B_1 = B_2 = B_3 = 2$ $B_4 = B_5 = 3$
19	4	8JS ($WS_1 \to WS_2 \to WS_3 \to WS_2 \to WS_3 \to WS_4 \to WS_3 \to WS_4$)	$B_i = 3, i = 1, 2, 3, 4$
20	5	8JS ($WS_1 \to WS_2 \to WS_3 \to WS_2 \to WS_3 \to WS_4 \to WS_5 \to WS_3$)	$B_i = 3, i = 1, 2, \ldots, 5$

Table 4.2: Comparison of the numbers of random switches (R.S.) and decision variables (D.V.) for the CRL configurations of Table 4.1 that result from (i) no refinement, (ii) Algorithm 2 (general refinement), and (iii) Algorithm 3 (CRL-customized refinement)

Conf.	Tangible Markings	No Refinement		Algorithm 2		Algorithm 3	
		Num. of R.S.	Num. of D.V.	Num. of R.S.	Num. of D.V.	Num. of R.S.	Num. of D.V.
1	19	20	27	5	5	7	7
2	7	4	4	1	1	1	1
3	33	40	56	11	11	16	16
4	87	128	177	35	35	46	46
5	579	1,007	1,374	269	269	331	331
6	42	71	84	9	9	12	12
7	148	346	463	49	49	67	67
8	301	742	966	112	112	149	149
9	1,593	4,304	5,498	677	677	849	849
10	4,245	13,302	20,948	2,083	2,290	2,732	3,000
11	2,511	7,573	11,368	1,513	1,513	1,817	1,817
12	1,162	2,781	4,018	678	678	734	734
13	1,045	2,468	3,759	609	609	765	765
14	261	519	693	106	106	124	124
15	1,518	4,256	5,887	759	759	811	811
16	694	1,851	2,534	243	243	258	258
17	41,097	163,695	270,738	30,805	35,420	38,653	44,755
18	20,389	74,655	109,948	12,313	12,313	17,047	17,047
19	98,133	322,052	525,166	80,142	85,117	88,479	94,034
20	198,231	788,731	1,270,562	139,496	154,069	186,617	208,312

3 in terms of the attained reduction of the numbers of random switches and decision variables. However, due to its structural nature, Algorithm 3 is computationally lighter, especially in the context of the larger CRL configurations. Also, as remarked in the previous section, Algorithm 3 can be applied in combination with Algorithm 2 in the following way: for any reachable Π_2-conditional vanishing marking M, Algorithm 3 will be first applied to a "crude" set $\mathcal{E}_u^{\Pi_2}(M)$ as a "filter". Then, thanks to the refinement that is effected by Algorithm 3, the corresponding random switch becomes smaller before computing the Π_2-untimed tangible reach for M for the eventual application of Algorithm 2.

4.1.6 The policy space $\hat{\Pi}_3$

For any refined policy space $\hat{\Pi}_2$, let us define the policy space $\hat{\Pi}_3$ as the subset of $\hat{\Pi}_2$ that is induced by the application of the additional constraint (6) in Section 3.1.2. Then, as the randomization factor δ goes to zero, the difference in the performance potential between the policy spaces $\hat{\Pi}_2$ and $\hat{\Pi}_3$ becomes negligible. Thus, the policy space $\hat{\Pi}_3$ will be the policy space adopted in the rest of this document, with the understanding that the set of enabled untimed transitions at any given vanishing marking M is the corresponding set $\mathcal{E}_u^{\hat{\Pi}_2}(M)$. On the other hand, in the subsequent developments, we shall not pay particular attention to the specific refinement algorithm applied (i.e., Algorithm 2, Algorithm 3, or even their combination).

4.2 Static random switches

Even though some refinement that maintains the performance potential of the policy space is applied, the numbers of random switches and decision variables are still proportional to the size of the underlying state space, as we can observe in Table 4.2. Furthermore, as the proof of Proposition 4 has established, the considered optimization problem can be seen as a semi-Markov decision process, with its state space containing all the tangible and all the vanishing markings in $\mathcal{R}^{\Pi_2}$ (or $\mathcal{R}^{\hat{\Pi}_2}$), and with the action space being defined by the set of all the tangible markings. Appendix A also shows how to define a simpler MDP for the RAS performance optimization problem, and the MDP obtained through this method provides a tight lower-bound to the number of decision variables of the refined policy space that maintains the performance potential.

The above remarks reveal that the refinement process alone is not enough for effectively controlling the complexity of the underlying MP formulation, and therefore, we must resort to approximation. More specifically, an intuitive way to reduce the number of the primary decision variables involved, is by making the probability

distributions of the random switches independent from the corresponding vanishing markings. In this way, the number of decision variables involved will not be proportional to the size of the corresponding state space. A possible realization of this independence is by setting the probability distributions of the various random switches dependent only upon the sets of the enabled untimed transitions, and not the vanishing markings themselves. In the relevant PN literature, this type of random switches is characterized as *static* [100, 20].

4.2.1 Applying static random switches

The notion of the "static random switch" can be formally applied in the MP formulation (9)–(13) by adding to the policy space $\hat{\Pi}_3$ the constraint:

$$Z_M = Z_{M'},\ \forall M, M' \text{ with } \mathcal{E}_u^{\hat{\Pi}_2}(M) = \mathcal{E}_u^{\hat{\Pi}_2}(M') \tag{27}$$

The resulting policy space will be denoted by $\hat{\Pi}_3^S$ and, as it will be revealed in the following, it admits a much more parsimonious representation than the policy space $\hat{\Pi}_3$.

More specifically, in the corresponding implementation of the formulation (14)–(16), no extra constraints are needed, but we should re-define the variable vector $\bar{\zeta}$ and the corresponding index-searching function as follows: Let Ξ be the set of all the possible enabling patterns of the random switches in the refined policy space $\hat{\Pi}_2$, i.e., $\Xi \equiv \{\mathcal{E} \subseteq \mathcal{T}_u : |\mathcal{E}| \geq 2 \wedge \exists M \in \mathcal{R}_{\mathcal{V}}^{\hat{\Pi}_2} \text{ s.t. } \mathcal{E}_u^{\hat{\Pi}_2}(M) = \mathcal{E}\}$. Then, $\bar{\zeta} \in \mathbb{R}_+^{\sum_{\mathcal{E}\in\Xi} |\mathcal{E}|-|\Xi|}$. Also, the corresponding index-searching function is defined as: $idx^S : \Xi \times \mathcal{T}_u \mapsto \mathbb{Z}_+ \cup \{\texttt{NULL}\}$. For the input $\mathcal{E} \in \Xi$ and $t \in \mathcal{T}_u$, if $t \in \mathcal{E}$ and t is not the last element in $\mathcal{E}$ according to lexicographic order, then $idx^S(\mathcal{E}, t)$ returns the index number i in the vector $\bar{\zeta}$ such that $\bar{\zeta}[i]$ is the probability that t is fired in the static random switch with enabling pattern $\mathcal{E}$; otherwise, it returns `NULL`. On the other hand, in the context of static random switches, we also retain the index-searching function $idx(\cdot)$ that was defined in Section 3.3, with its original domain $\mathcal{R}_{\mathcal{V}}^{\hat{\Pi}_2} \times \mathcal{T}_u$, but with the further understanding

that $idx(M,t) \equiv idx^S(\mathcal{E}_u^{\hat{\Pi}_2}(M),t)$. Finally, the objective function should also be changed accordingly to reflect the new set of the refined decision variables. But in the following, we shall still apply the notation $\eta(\bar{\zeta})$ for the objective function.

Unlike the refined policy space $\hat{\Pi}_2$ introduced in the previous section, the policy space $\hat{\Pi}_3^S$ does not maintain the performance potential of $\hat{\Pi}_3$. However, we can observe that $\hat{\Pi}_3^S$ still contains a large number of policies. More specifically, when $\delta \downarrow 0$, $\hat{\Pi}_3^S$ contains all the static-priority policies,[6] but it adds the flexibility that the priorities assigned by these policies may change when different sets of untimed transitions are enabled. In the context of the MDP-based modeling of the considered optimization problem that was discussed at the beginning of this section, the policy space $\hat{\Pi}_3^S$ essentially defines an *aggregation* of the underlying state space, by enforcing a correlation of the decisions that are made at states that belong to the same state aggregate.

The worst-case complexity in terms of the number of the primary decision variables that are employed by the policy space $\hat{\Pi}_3^S$, is still super-polynomial. More specifically, the vector $\bar{\zeta}$ has the dimension of $\sum_{\mathcal{E}\in\Xi}|\mathcal{E}|-|\Xi|$, with a worst possible value of $O(2^{|\mathcal{T}_u|})$. But in the more practical application context of RAS, it is rare that all the processing stages of all the process types compete for the same resource. Therefore, since the refinement in Section 4.1 maintains only conflicting options in any employed random switch,[7] the empirical representational complexity of the policy space $\hat{\Pi}_3^S$ can be quite benign.

[6]Static-priority policies are defined as the policies that set priorities to each processing stage in the RAS, or to each untimed transition in the case of the RAS-modeling GSPN. Note that some widely applied heuristics in the industrial practice, such as first-buffer-first-serve (FBFS) and last-buffer-first-serve (LBFS), are all static priority policies.

[7]It is also possible that some processing stages may not require the same resource type, but they are actually in competition due to some deadlock avoidance consideration. This issue may make the set of static random switches a bit larger than all the possible combinations of resource competition patterns (i.e., the possible combinations of processing stages that are in conflict for the various resource types), but still much smaller than the power set of $\mathcal{T}_u$.

4.2.2 A numerical experiment for the application of static random switches

The empirical representational complexity for the policy space $\hat{\Pi}_3^S$, that employs the static random switch concept, is demonstrated with some numerical results that are obtained from the same testbed that was used in the previous section. Table 4.3 shows the reductions in the numbers of random switches and decision variables that are attained by redefining the decision variables with respect to the static random switches of the 20 CRL configurations of Table 4.1. The table also lists, for comparison purposes, the numbers of random switches and decision variables of the original policy space Π_2, which have already been reported in Table 4.2. Then, in the remaining columns the table reports the numbers of the static random switches and their corresponding decision variables under three different refinement settings: no refinement, general refinement with Algorithm 2, and CRL-customized refinement with Algorithm 3.

The reader may observe that the complexity reduction caused by the application of static random switches is more significant in larger state spaces, particularly when combined with the refinement process of Section 4.1.

Another interesting observation concerns the comparison of the numbers of the decision variables that result by employing static random switches under the two refinement processes that were introduced in the previous sections. The CRL-customized refinement process (Algorithm 3) leads to a higher reduction of the numbers of the employed random switches and decision variables than the reduction that is effected by the general process (Algorithm 2), since it can correctly identify the fundamental conflicts in the allocation of servers and buffer slots in CRL, and it aggregates effectively conflicts of similar type appearing in different markings. On the other hand, the general refinement process, by picking the set of the enabled untimed transitions in lexicographic order, fails to pronounce the aforementioned essential conflicts, and this is reflected in the eventual aggregation that is defined by the corresponding policy

Table 4.3: Comparison of the numbers of random switches (R.S.) and decision variables (D.V.) for the CRL configurations of Table 4.1 when applying the static random switches

Conf.	No static random switches		Applying static random switches					
			No refinement		Algorithm 2		Algorithm 3	
	Num. of R.S.	Num. of D.V.	Num. of R.S.	Num. of D.V.	Num. of R.S.	Num. of D.V.	Num. of R.S.	Num. of D.V.
1	20	27	11	16	2	2	2	2
2	4	4	4	4	1	1	1	1
3	40	56	12	18	2	2	2	2
4	128	177	12	18	2	2	2	2
5	1,007	1,374	12	18	2	2	2	2
6	71	84	20	27	1	1	1	1
7	346	463	35	61	2	2	2	2
8	742	966	35	61	2	2	2	2
9	4,304	5,498	38	68	2	2	2	2
10	13,302	20,948	397	1,104	13	15	10	12
11	7,573	11,368	113	251	4	4	4	4
12	2,781	4,018	89	181	4	4	4	4
13	2,468	3,759	81	165	5	5	4	4
14	519	693	73	136	5	5	4	4
15	4,256	5,887	188	414	6	6	6	6
16	1,851	2,534	164	342	6	6	6	6
17	163,695	270,738	579	1,827	15	17	10	12
18	74,655	109,948	191	497	4	4	4	4
19	322,052	525,166	771	2,456	19	22	12	14
20	788,731	1,270,562	739	2,342	14	17	10	12

space $\hat{\Pi}_3^S$.

4.2.3 The potential performance losses incurred by the employment of static random switches and a plan for their retrieval

The application of static random switches reduces the representational and the computational complexity of the corresponding scheduling problem at the cost of compromising the optimality of this problem, or some of the operational efficiency of the underlying RAS. However, part of this cost can be retrieved in a controlled manner, through a partial disaggregation of (some of) the state aggregates that are defined by

the static random switches. This possibility defines an effectively manageable trade-off between the tractability of the pursued approximation and the sub-optimality of the derived solutions: at one extreme point of this spectrum, namely that of the complete aggregation that is defined by all the static random switches, the policy space is exactly $\hat{\Pi}_3^S$; on the other extreme point, corresponding to the complete disaggregation of all the static random switches, the policy space maintains the performance potential of the original policy space $\hat{\Pi}_3$.

From an operational standpoint, the aforementioned partial disaggregation can be effected by allowing the random switches with the same enabling pattern $\mathcal{E}$ to have different probability distributions, but correlate these distributions in some ways so that there are fewer degrees of freedom and a fewer number of decision variables for regulating these probability distributions. The correlated random switches in these intermediate cases will be called *partially disaggregated* random switches in the sequel. On the other hand, the random switches that correspond to the two aforementioned extreme cases will be respectively characterized as *static* and *dynamic*. In the next section, we present a random-switch representation for partial disaggregation that is based on the notion of "feature functions". And this representation has the potential to become a control mechanism for the trade-off between complexity and optimality.

4.3 An alternative representation for random switches that facilitates partial disaggregation

In order to accommodate increasing degrees of freedom for correlated random switches, the decision variables of the MP formulations of the considered scheduling problem should impact the selection probabilities that are regulated by random switches in a more indirect way. In this section, first we introduce such an indirect representation for static random switches, and then we employ this indirect representation in order to develop a partial disaggregation scheme. In the resulting policy spaces, the new

representation to be introduced in this section will be applied to any partially disaggregated random switches. However, the old representation that was introduced in Section 4.2 will still be used to represent those static random switches that are not selected to be disaggregated.

An indirect representation for static random switches The most common way to effect such a representation is by using a random switch that has the form of a *Gibbs measure* [50]. More specifically, consider a random switch (either static or non-static) of a vanishing marking M and let the corresponding vector of decision variables be $v \in \mathbb{R}^{|\mathcal{E}_u^{\hat{\Pi}_2}(M)|}$. In this new regime, the probability of firing an untimed transition $\hat{t} \in \mathcal{E}_u^{\hat{\Pi}_2}(M)$ is no longer the value of the decision variable $v[\hat{t}]$ itself, but the exponent of this value, divided by a normalization factor in order to establish a probability distribution; i.e.,

$$Z_M[\hat{t}] = \frac{\exp(v[\hat{t}])}{\sum_{t \in \mathcal{E}_u^{\hat{\Pi}_2}(M)} \exp(v[t])} \tag{28}$$

Since the definition of the selection probabilities through Equation (28) introduces naturally a certain degree of randomization for the corresponding random switches, we propose to remove from the formulation (14)–(16) the constraints (15) and (16) that correspond to these random switches.[8] Next, we discuss how we shall employ the concept of the Gibbs measure that is defined by Equation (28) in order to effect the proposed partial disaggregation scheme.

A feature-based partial disaggregation scheme Consider an enabling pattern $\mathcal{E} \subseteq \mathcal{T}_u$, and further suppose that either it is a static random switch, or it generates

[8] From a more practical standpoint, the elimination of the relevant randomization requirement that is expressed by the constraints (15) and (16) of formulation (14)–(16), is necessitated by the fact that the enforcement of these constraints for the partially disaggregated random switches in the computation scheme that is presented in Chapter 5, would be an intractable task; we shall return to this issue when we discuss the corresponding results in that chapter.

a set of correlated, partially disaggregated random switches.[9] The corresponding decision variables are encoded to the vector of decision variables $\bar{\zeta}$. Let $K \in \mathbb{Z}_+$ be a pre-determined parameter that controls the degree of disaggregation. Then, for each enabling pattern $\mathcal{E}$ that generates partially disaggregated random switches, $\bar{\zeta}$ will allow $|\mathcal{E}| \times K$ components, each of which is associated to a pair of an option in $\mathcal{E}$ and a feature function. An index-searching function $idx^{DA} : \mathcal{E} \times \mathcal{T}_u \mapsto \mathbb{Z}_+^K$ is applied to locate the indices of these components. For the input $\mathcal{E} \in \Xi$ and $\hat{t} \in \mathcal{T}_u$, if $\hat{t} \notin \mathcal{E}$, then $idx^{DA}(\mathcal{E}, \hat{t}) = \texttt{NULL}$; otherwise, $idx^{DA}(\mathcal{E}, \hat{t})$ returns the index numbers of the K components in the vector $\bar{\zeta}$ corresponding to $\hat{t}$ in $\mathcal{E}$ (note that for some $\mathcal{E}' \neq \mathcal{E}$, it is possible that $\hat{t} \in \mathcal{E}'$, but this time $idx^{DA}(\mathcal{E}', \hat{t})$ will return a different set of K index numbers). These K components of $\bar{\zeta}$ form a set of tunable parameters for a particular option $\hat{t} \in \mathcal{E}$. More specifically, there is a set Ψ of K feature functions, i.e., $\Psi \equiv \{\psi_k(\cdot) : \mathcal{R}_{\mathcal{V}} \mapsto \mathbb{R}, k = 1, \ldots, K\}$. The feature functions are linearly independent with respect to each other in the vanishing state space. In other words, for any particular function $\psi_{\hat{k}}(\cdot)$, there does not exist a set of real coefficients $c_k, k = 1, \ldots, K$, such that for any vanishing marking $M \in \mathcal{R}_{\mathcal{V}}^{\hat{\Pi}_2}$, $\psi_{\hat{k}}(M) = \sum_{k \neq \hat{k}} c_k \psi_k(M)$. Then, we can express the probability of firing a particular untimed transition $\hat{t} \in \mathcal{E}$ at a vanishing marking M, when $\mathcal{E} = \mathcal{E}_u^{\hat{\Pi}_2}(M)$ and the corresponding random switch is partially disaggregated, as follows:

$$Z_M[\hat{t}] = \frac{\exp\Big(\sum_{k=1}^{K} \psi_k(M) \cdot \bar{\zeta}\big[idx^{DA}(\mathcal{E}, \hat{t})[k]\big]\Big)}{\sum_{t \in \mathcal{E}} \exp\Big(\sum_{k=1}^{K} \psi_k(M) \cdot \bar{\zeta}\big[idx^{DA}(\mathcal{E}, t)[k]\big]\Big)} \tag{29}$$

For a given GSPN with state space $\mathcal{R}^{\hat{\Pi}_2} = \mathcal{R}_{\mathcal{V}}^{\hat{\Pi}_2} \cup \mathcal{R}_{\mathcal{T}}^{\hat{\Pi}_2}$, as $K \uparrow |\mathcal{R}_{\mathcal{V}}^{\hat{\Pi}_2}|$ and the functions $\psi_k(\cdot), k = 1, \ldots, K$, maintain their linear independence, the aforementioned mechanism of random switches tends to give the probability distributions for the various vanishing markings that are linearly independent vectors. In particular, if

[9] The hybrid of the two types for one enabling pattern is possible, but the allocation of $\bar{\zeta}$ and the index-searching function will not be the same as in this paragraph. In the current research work, only non-mixed enabling patterns are considered.

$K = |\mathcal{R}_{\mathcal{V}}^{\hat{\Pi}_2}|$ and the feature functions are the corresponding indicator functions for each vanishing marking, then the variable $\bar{\zeta}\big[idx^{DA}(\mathcal{E}, t)[k]\big]$ in Equation (29) is effective only if $\psi_k(\cdot)$ is the indicator function for the current marking M and it plays the role of the variable $v[t]$ in Equation (28). More generally, the degrees of freedom between the static and dynamic random switches can be adjusted by controlling the number and the content of the feature functions.

It is clear from the above discussion that the Gibbs measure materialized by Equation (29) constitutes a general representation mechanism for policy spaces that are defined through a combination of static and partially disaggregated random switches. But the complete exploitation of this mechanism in the considered problem context needs the systematic investigation of a host of additional issues. For instance, we need to specify criteria that will lead the selection of the enabling pattern(s) to be disaggregated. These criteria could be either static or dynamic, i.e., either taking into consideration only certain attributes of the underlying RAS, or further considering the structure of the optimized policies that are computed for a sequence of previously defined policy spaces. Some additional issues include the development of the algorithms that will determine the optimized policies in the context of the mixed policy spaces that were defined above, and the selection of a good set of feature functions that will also provide sufficient levels of linear independence along the line that was discussed in the previous paragraphs. We shall address some of these issues in the next chapter where we discuss the broader problem of solving the MP formulation for the randomized policy spaces that were defined in this section, through stochastic approximation algorithms.

CHAPTER V

SOLVING THE FORMULATED SCHEDULING PROBLEM WITH STOCHASTIC APPROXIMATION

As mentioned in the previous parts of this document, it is hard to estimate the value of the objective function for the MP formulation (14)–(16) when the size of the considered RAS increases, since the size of the underlying state space and the corresponding vector of the steady-state distribution become very large. The methods presented in the previous chapter only control the representational complexity of the solution space, in terms of the number of decision variables, to a tractable level; but the objective function itself is still hard to evaluate. Furthermore, due to the coupling effects on the random switches that were introduced by some of the complexity control methods of Chapter 4, the classical MDP modeling method of Appendix A is no longer applicable. Additional substantial increases of the state space of the stochastic processes involved can result from the phase-type approximation of any general timing distributions that might appear in the RAS-modeling GSPN (c.f. Section 2.3) since the corresponding Markovian subnets will enlarge the GSPN structure. To cope with all the aforementioned difficulties, in this chapter we resort to simulation optimization in order to solve the MP formulations that were derived in the previous chapters. More specifically, since the objective function is differentiable and all the decision variables are continuous, the method of stochastic approximation (SA) is appropriate for the considered simulation optimization problem, and this is the method to be employed in the subsequent developments.

Hence, in this chapter, first we shall introduce the basic framework of the SA

algorithms, and some general conditions for the asymptotic convergence of these algorithms. Next, the general framework of the SA algorithms will be customized to the performance optimization problem that was formulated in the previous chapters. In particular, we shall develop the necessary formulae for the calculation of the improvement directions to be employed by the sought algorithm(s), and we shall establish the aforementioned conditions for asymptotic convergence. Furthermore, since the policy spaces to be employed by the considered algorithms are only implicitly defined by the underlying GSPN structure, the applied DAP, and the remaining operational conditions that characterize these policy spaces, an additional important task to be supported by the sought algorithms is (i) the detection of the particular structure that defines the considered policy spaces, in terms of the employed random switches and the corresponding decision variables, through the performed simulation, and (ii) the encoding of the identified structure by means of some pertinent data structures that are defined and maintained during the algorithm execution. The second part of the current chapter will address this algorithmic functionality, as well. In the third part of the chapter, the developed SA algorithms will also be considered from the more practical standpoint of their transient behavior, and we shall seek to enhance the corresponding performance through some results drawn from the area of statistical inference. Finally, we also address the necessary extensions so that the developed algorithms are applicable in the setting of the partially disaggregated random switches discussed in Section 4.3.

5.1 The basic stochastic approximation framework

Consider a general nonlinear optimization problem

$$\max_{\bar{\zeta} \in H} \eta(\bar{\zeta}) \tag{30}$$

where the feasible region $H \subseteq \mathbb{R}^p$. Furthermore, let us assume that the evaluation of the function $\eta(\bar{\zeta})$ for any given value $\bar{\zeta}$ is not practically tractable; in particular, in

the sequel we shall assume that there exists a random variable X whose probability distribution depends on the parameters specified by $\bar{\zeta}$, and $\eta(\bar{\zeta}) = \mathbb{E}[X|\bar{\zeta}]$. Then, a stochastic approximation (SA) algorithm is essentially a search process for a local optimum of the aforementioned optimization problem that is based on the following recursion [53]:

$$\bar{\zeta}_{n+1} = \texttt{proj}_H\left(\bar{\zeta}_n + \gamma_n Y_n\right), n \geq 0 \tag{31}$$

In the recursion of Equation (31), $\bar{\zeta}_n$ and $\bar{\zeta}_{n+1}$ are the values of the problem decision variables at iterations n and $n+1$, respectively. In particular, $\bar{\zeta}_0$ is the *initial solution* for the SA algorithm, which can be either randomly sampled from the feasible region H, or determined through some heuristics. γ_n is a non-negative and deterministic scalar, called the *step size* at iteration n.[1] Y_n is a random vector that together with the step size γ_n determine the perturbation that is added to the original variable vector $\bar{\zeta}_n$ in order to obtain the new variable vector $\bar{\zeta}_{n+1}$. In the considered problem context where $\eta(\bar{\zeta}) = \mathbb{E}[X|\bar{\zeta}]$, Y_n will be obtained from the simulation of a stochastic process that can determine the statistics of the random variable X with parameter $\bar{\zeta}_n$. Finally, $\texttt{proj}_H(\cdot) : \mathbb{R}^p \mapsto H$ is the *projection* operator of Euclidean norm that keeps $\bar{\zeta}_n$ always in the feasible region H; more specifically, for any vector $v \in \mathbb{R}^p$, $\texttt{proj}_H(v) \equiv \arg\min_{v' \in H} \sum_{k=1}^{p} (v'[k] - v[k])^2$.

As observed from the recursion (31), the dynamics of the stochastic process $\{\bar{\zeta}_n\}, n = 0, 1, \ldots$ is guided by the sequence $\{Y_n\}$. Intuitively, we expect that some movement along the direction of Y_n can incur some improvement in terms of $\eta(\bar{\zeta}_n)$ in the optimization problem (30). Hence, Y_n is typically called the *improvement direction*. If the objective function $\eta(\cdot)$ is differentiable, or more formally, the optimization problem satisfies Condition 5 below, then, for the maximization problem (30), Y_n is

[1] It is also possible that γ_n is determined not only by n and the initial input of the algorithm but also by some information from the previous iterations. In this case, γ_n is a random variable. Then, the convergence condition of (31) is a little different from the deterministic case. More discussions on the different settings of the step size can be found in Chapter 17 of [27]. However, in this document, only deterministic step sizes are considered.

often an estimator of $\nabla\eta(\bar{\zeta})$.

Condition 5 *The objective function $\eta : H \mapsto \mathbb{R}$ is a continuously differentiable real-valued function.*

Under Condition 5, once the value of $\bar{\zeta}$ reaches a (local) maximum, the improvement direction Y_n with expectation $\nabla\eta(\bar{\zeta})$ cannot incur any further improvement, and the algorithm should terminate. Obviously, if $H = \mathbb{R}^p$, then, any local optimum must be sought among the values $\bar{\zeta}^*$ with $\nabla\eta(\bar{\zeta}^*) = 0$. On the other hand, it is also possible that $\nabla\eta(\bar{\zeta}') \neq 0$ at some solution point $\bar{\zeta}'$, but the restriction imposed by H does not allow any improvement along the direction of $\nabla\eta(\bar{\zeta}')$ (or more precisely, any directions that are in acute angle with the vector $\nabla\eta(\bar{\zeta}')$). In the latter case, $\bar{\zeta}'$ is also a local optimum, but the corresponding condition on the vector $\nabla\eta(\bar{\zeta}')$ is not as simple as in the case where $H = \mathbb{R}^p$. However, some pertinent condition can be derived if H satisfies the following condition:

Condition 6 *H is connected, compact and non-empty, and it is defined by $H \equiv \{\bar{\zeta} \in \mathbb{R}^p : c_j(\bar{\zeta}) \leq 0, j = 1, \ldots, l_c\}$, where each constraint function $c_j(\cdot)$ is a continuously differentiable real-valued function. Furthermore, when a constraint function is binding, its gradient is non-zero, i.e., $\forall j$, $\nabla c(\bar{\zeta}) \neq 0$ if $c_j(\bar{\zeta}) = 0$.*

Given the satisfaction of Condition 6, the vectors $\nabla\eta(\bar{\zeta}^*)$ corresponding to some (constrained or unconstrained) local optimum $\bar{\zeta}^*$ are characterized by the following condition which is an adaptation of the famous Karush-Kuhn-Tucker (KKT) condition of the nonlinear optimization theory [67] to the considered problem context:

Condition 7 *Let $B(\bar{\zeta}^*) \equiv \{c_j(\cdot) : c_j(\bar{\zeta}^*) = 0\}$ be the set of the binding constraint functions at $\bar{\zeta}^*$. Then there exists a set of* non-negative *scalars λ_j, $j \in B(\bar{\zeta}^*)$, such that*

$$\nabla\eta(\bar{\zeta}^*) = \sum_{j \in B(\bar{\zeta}^*)} \lambda_j \nabla c_j(\bar{\zeta}^*) \tag{32}$$

The scalars $\lambda_j,\ j \in B(\bar{\zeta}^*)$, appearing in Condition 7 above, are called *Lagrange multipliers*. Condition 7 can be explained in the following intuitive way. Consider a solution point $\bar{\zeta}^*$, where the gradient $\nabla\eta(\bar{\zeta}^*)$ is either zero, or can be represented as a conic combination of the vectors $\nabla c_j(\bar{\zeta}^*)$, $j \in B(\bar{\zeta}^*)$. Furthermore, notice that any movement along the direction of $\nabla c_j(\bar{\zeta}^*)$, for any $j \in B(\bar{\zeta}^*)$, will cause the violation of the constraint $c_j(\bar{\zeta}) \leq 0$. Therefore, if Condition 7 is satisfied, we cannot move $\bar{\zeta}^*$ along any directions that form an acute angle with the gradient vector $\nabla\eta(\bar{\zeta}^*)$ and keep the solution point $\bar{\zeta}^*$ in the feasible region H. Hence, $\bar{\zeta}^*$ can be declared as a local maximum since there are no points nearby with higher η value.

In the sequel, we are interested in implementations of the SA algorithm of Equation (31) on the optimization problem (30) that can make the stochastic process $\{\bar{\zeta}_n\}$ move towards some local maxima that satisfy Condition 7 with probability 1, as $n \to \infty$. And these dynamics are characterized as *asymptotic convergence* of the SA algorithm.

In [53], several sets of conditions for asymptotic convergence were introduced. In the next section, we present one set of such conditions. This condition set will be employed later, in Section 5.2, for establishing the asymptotic convergence of the SA algorithm that we shall define for the solution of the MP formulation (14)–(16), which is the focus of this work.

5.1.1 An asymptotic convergence result for the considered SA algorithm

In this section first we introduce an additional set of conditions that qualify further the optimization problem (30) and the recursion (31), and subsequently, we present a theorem that establishes the asymptotic convergence of the corresponding SA algorithm.

The next two conditions can be considered as stronger versions of Conditions 5 and 6.

Condition 8 *The objective function $\eta : H \mapsto \mathbb{R}$ is a twice continuously differentiable real-valued function.*

Condition 9 *Condition 6 holds, and each constraint function $c_j(\cdot)$ is a twice continuously differentiable real-valued function.*

On the other hand, the next condition qualifies further the feasible region of the optimization problem (30).

Condition 10 *The set of all feasible solutions that satisfy Condition 7, denoted by S_H, can be partitioned into disjoint compact and connected subsets S_j, $j = 0, 1, \ldots$*

Finally, there are also some requirements on the step size γ_n and the improvement direction Y_n of the SA recursion (31).

Condition 11 $\gamma_n \geq 0, n = 0, 1, \ldots$, $\sum_{n=0}^{\infty} \gamma_n = \infty$, *and* $\sum_{n=0}^{\infty} \gamma_n^2 < \infty$.

Condition 12 *The expectation of the Euclidean norm of each Y_n is bounded, i.e.,*

$$\sup_n \mathbb{E}[\sum_{k=1}^{p} Y_n[k]^2] < \infty$$

Condition 13 *Condition 5 holds, and $\{Y_n\}$ is a stochastic process such that*

$$\mathbb{E}[Y_n | \bar{\zeta}_0, Y_i, i < n] = \nabla \eta(\bar{\zeta}_n) + \beta_n$$

and $\lim_{n \to \infty} \beta_n = 0$ *with probability 1.*

Given the aforementioned conditions, the next theorem establishes the asymptotic convergence of recursion (31):

Theorem 3 *[53] If Conditions 8–13 hold, then, as $n \to \infty$, the sequence $\{\bar{\zeta}_n\}$ generated by recursion* (31) *converges with probability 1 to a unique subset S_j of the set S_H that was defined in Condition 10.*

In the next section, we will customize the basic SA algorithm that was described above to the performance optimization problem that was developed in the previous chapters.

5.2 Customizing the basic SA algorithm to the considered performance optimization problem

The key components of the optimization problem (30) and the SA recursion (31) include: the decision variables $\bar{\zeta}_n$, the feasible region H, the objective function $\eta(\bar{\zeta})$, the improvement direction Y_n, the step size γ_n, and the projection operator $\texttt{proj}_H(\cdot)$. On the other hand, the asymptotic convergence of the recursion (31) is based on the further qualification of those components through additional conditions, like the conditions involved in the statement of Theorem 3. In this section, we shall customize the SA recursion (31) in order to develop an asymptotically convergent SA algorithm for the performance optimization problem considered in this work, where the latter is defined by the uniformized version of the MP formulation (14)–(16). Hence, the following subsections provide a detailed description of the considered problem and the key ingredients of the proposed SA Algorithm. The section concludes with a formal statement of the proposed algorithm and its asymptotic convergence.

5.2.1 The considered optimization problem

The particular optimization problem that is considered in this section is the maximization of the steady-state average reward of the DTMC $\hat{\mathcal{M}}$ that results from the uniformizing procedure that was discussed in Section 3.3. Using the notation that was introduced in the previous chapters, this maximization problem can be formally stated as follows:

$$\text{maximize} \quad \eta(\bar{\zeta}) = \pi^T \cdot \hat{r} \tag{33}$$

$$\text{subject to} \quad \sum_{t:idx(M,t)\neq \texttt{NULL}} \bar{\zeta}[t] \leq 1 - \frac{\delta}{|\mathcal{E}_u^{\hat{\Pi}_2}(M)|} \qquad \forall M \in \mathcal{R}_{\mathcal{V}}^{\hat{\Pi}_2} : |\mathcal{E}_u^{\hat{\Pi}_2}(M)| \geq 2 \tag{34}$$

$$\bar{\zeta}[idx(M,t)] \geq \frac{\delta}{|\mathcal{E}_u^{\hat{\Pi}_2}(M)|} \quad \forall M \in \mathcal{R}_{\mathcal{V}}^{\hat{\Pi}_2},\ \forall t \in \mathcal{T}_u : idx(M,t) \neq \texttt{NULL} \tag{35}$$

We remind the reader that in the above formulation, the decision variables $\bar{\zeta}$ define the selection probabilities of the various (dynamic or static) random switches.

The constraints (34) and (35) define the feasible region H in terms of the simplices that correspond to these random switches. Hence, H is a closed polytope, and therefore, connected, bounded and non-empty. Constraints (34) and (35) also define the constraint functions c_j that appear in Condition 6 of Section 5.1.1, as some affine functions of the vector $\bar{\zeta}$. As a result, all the constraint functions c_j of the considered formulation are twice continuously differentiable. The next lemma results immediately from the previous remarks.

Lemma 2 *The feasible region H defined by Equations* (34) *and* (35) *satisfies Condition 9 in Section 5.1.1.*

Remark We also notice, for completeness, that if the alternative representation in Section 4.3 is applied to some random switches, the feasible region H will become unbounded with respect to the corresponding components. Then, H is no longer compact, and Conditions 6 and 9 will be no longer satisfied. We shall return to this issue in Section 5.4.1 where we address the relevant variation of the considered optimization problem. □

Next, let us focus on the objective function $\eta(\bar{\zeta})$ of the considered MP formulation, which is defined in Equation (33). More specifically, Equation (33) defines $\eta(\bar{\zeta})$ as the inner product of the steady-state distribution π of the corresponding DTMC $\hat{\mathcal{M}}$ and the vector of immediate rewards $\hat{r}$, which is a constant vector.[2] Thus, $\eta(\bar{\zeta})$ is essentially a weighted sum of the components of the vector π.

On the other hand, the steady-state distribution vector π is a set of "auxiliary" variables that depends on the value of $\bar{\zeta}$. More specifically, π is the solution of the

[2]The reader is referred to Section 2.3.3 for the initial definition of the vector $\hat{r}$, and to Section 3.3 for the final modification of this vector under the applied uniformization.

following linear system of equations:

$$\begin{aligned} \pi^T P &= \mathbf{1}^T \\ \pi^T \mathbf{1} &= 1 \end{aligned} \tag{36}$$

In Equation (36), P is the transition probability matrix of the DTMC $\hat{\mathcal{M}}$ that depends on the value of $\bar{\zeta}$. The CTMC-construction procedures that were presented at the end of Section 2.3.3, together with the uniformization procedure that was discussed in Section 3.3, imply that each element of the matrix P is a polynomial function of the selection probabilities of the employed random switches, and thus, of the decision variables $\bar{\zeta}$.

Furthermore, π is well-defined for any $\bar{\zeta}$ in the randomized policy spaces that are considered in this chapter, since it is known to be equal to the steady-state probability of the corresponding CTMC $\mathcal{M}$ that models the continuous-time dynamics of the RAS-modeling GSPN (c.f. the paragraphs on unformization in Section 3.3). But then, Equation (36) implies that each component of π can be expressed as a rational function (i.e., the ratio of two polynomials) of $\bar{\zeta}$. And this remark further applies to the objective function of Equation (33), since it is a weighted sum of the aforementioned functions. But then, we have the following result:

Lemma 3 *The objective function $\eta(\bar{\zeta})$ defined by Equation* (33) *is twice continuously differentiable in the feasible region H defined by Equations* (34)–(35).

Next we argue that the problem formulation (33)–(35) satisfies also the Condition 10 of Section 5.1.1. Indeed, every local optimum of $\eta(\bar{\zeta})$ in the interior of H will be located on a "plateau" of this continuously differentiable function which constitutes a compact and connected set. On the other hand, the regions that might contain constrained local optima for the considered optimization problem are defined by the intersections of the objective function $\eta(\bar{\zeta})$ with some of the facets of the polyhedron that corresponds to the feasible region H. Therefore, they will also be compact and

connected sets. The next lemma gives a formal expression to these remarks.

Lemma 4 *The optimization problem defined by Equations* (33)–(35) *satisfies Condition 10 in Section 5.1.1.*

Up to this point, we have concretized the main MP formulation that is addressed in this chapter, and we have established that it possesses the key properties that are required by the Conditions 8, 9 and 10 of Section 5.1.1. Next, we turn to the most important ingredient of the SA recursion (31): the specification and computation of the improvement direction Y_n in a way that also satisfies Conditions 12 and 13 of Section 5.1.1.

5.2.2 Determining the improvement direction

The general problem of estimating the gradient of a function $\eta(\bar{\zeta}) = E[X|\bar{\zeta}]$ with respect to the parameter vector $\bar{\zeta}$ through simulation-based methods is addressed in [35]. That work overviews a number of methods that have been proposed for this problem, and classifies them into *direct* and *indirect*.

Indirect methods, also characterized as "black-box" methods, estimate the considered gradient from the basic definition of this concept, by generating estimates of the objective function at appropriately perturbed values of the vector $\bar{\zeta}$. Typically, these methods are characterized by a large variance and they are computationally costly. The work of [98] proposes some variance reduction techniques that are appropriate for these methods and they are based on the notion of the simultaneous perturbations stochastic approximation (SPSA). But, in general, indirect methods are usually applied when little information is known about the stochastic process that determines the objective function $\eta(\bar{\zeta})$.

On the other hand, direct methods can control better the variance of the corresponding estimators, but they require additional information and structure for the underlying stochastic process that determines the objective function. Since the timed

dynamics that are considered in this chapter can be modeled by the ergodic DTMC $\hat{\mathcal{M}}$ with computable transition probabilities, the direct methods of [35, 36] seem more appropriate. In the rest of this section, we focus on a particular such method that is known as the method of *likelihood ratio (LR)* estimation [4]. This method is enabled by the regenerative nature of the aforementioned DTMC $\hat{\mathcal{M}}$, and it can provide an asymptotically unbiased estimator for the sought gradient.

Likelihood-ratio-based gradient estimator In order to establish the sought estimator of the gradient of the objective function $\eta(\bar{\zeta})$ that is defined by Equation (33), let us first pick arbitrarily a tangible marking M^* from the state space $\mathcal{R}_{\mathcal{T}}^{\hat{\Pi}_2}$ of the finite-state ergodic DTMC $\hat{\mathcal{M}}$ mentioned in Section 5.2.1; in the sequel we shall refer to this marking as the *reference marking*. Furthermore, consider a sample path $\{X(t) : t = 0, 1, \ldots\}$ of the DTMC $\hat{\mathcal{M}}$ starting from marking M^* at time period $t = 0$. Then, due to the ergodicity of $\hat{\mathcal{M}}$, the marking M^* will be visited in this sample path for an infinite number of times. Let us also define the stopping times

$$\tau_k \equiv \begin{cases} 0 & \text{if } k = 0 \\ \inf\{t > \tau_{k-1} : X(t) = M^*\} & \text{if } k = 1, 2, \ldots \end{cases} \tag{37}$$

Due to the memoryless property of Markov chains, each visit to M^* can be seen as a "regeneration point" of the stochastic process. Therefore, for any $j = 1, 2, \ldots$, the sample path segments $\{X(\tau_{j-1}) = M^*, X(\tau_{j-1} + 1), \ldots, X(\tau_j - 1)\}$ are called *regenerative cycles*; these segments are independent from each other and can be seen as individual sample paths. Furthermore, the steady-state average reward of process $\hat{\mathcal{M}}$ can be expressed as a ratio of two expectations that are based on the regenerative cycles mentioned above:

$$\eta(\bar{\zeta}) = \frac{\mathbb{E}\left[\sum_{t=0}^{\tau_1 - 1} \hat{r}[X(t)] | \bar{\zeta}, M^*\right]}{\mathbb{E}[\tau_1 | \bar{\zeta}, M^*]} \tag{38}$$

In the sequel, we will suppress the dependence on the decision variables $\bar{\zeta}$ whenever there is no risk of confusion. Also, in order to facilitate the subsequent developments,

we introduce the following notation:

$$\begin{aligned} z(\hat{r}) &\equiv \mathbb{E}[\sum_{t=\tau_{j-1}}^{\tau_j-1} \hat{r}[X(t)]|M^*] \\ z(\mathbf{1}) &\equiv \mathbb{E}[\sum_{t=\tau_{j-1}}^{\tau_j-1} 1|M^*] = \mathbb{E}[\tau_j - \tau_{j-1}|M^*] \end{aligned} \tag{39}$$

Then, from Equation (38) we have:

$$\begin{aligned} \nabla\eta &= \nabla\left(\frac{z(\hat{r})}{z(\mathbf{1})}\right) \\ &= \frac{z(\mathbf{1})\cdot\nabla z(\hat{r}) - z(\hat{r})\cdot\nabla z(\mathbf{1})}{z(\mathbf{1})^2} \end{aligned} \tag{40}$$

The expectations $z(\hat{r})$ and $z(\mathbf{1})$ can be estimated according to their definition in Equation (39). On the other hand, to estimate the gradients $\nabla z(\hat{r})$ and $\nabla z(\mathbf{1})$, the theory of the LR method for ergodic DTMCs [4] implies that [3]

$$\begin{aligned} \nabla z(\hat{r}) &= \mathbb{E}\left[\sum_{t=0}^{\tau_1-1} \hat{r}[X(t)] \sum_{l=1}^{t} \frac{\nabla P[X(l-1),X(l)]}{P[X(l-1),X(l)]} \mid M^*\right] \\ \nabla z(\mathbf{1}) &= \mathbb{E}\left[\sum_{t=0}^{\tau_1-1} \sum_{l=1}^{t} \frac{\nabla P[X(l-1),X(l)]}{P[X(l-1),X(l)]} \mid M^*\right] \end{aligned} \tag{41}$$

We remind the reader that P is the transition probability matrix of the uniformized DTMC $\hat{\mathcal{M}}$, and its elements are functions of the decision variables $\bar{\zeta}$. Furthermore, $\nabla P[x,y]$ denotes the gradient vector with respect to $\bar{\zeta}$ of the element $P[x,y]$ of the

[3] For a complete justification of the formulae in the right-hand-side of Equation (41), the reader is referred to [4]. However, a brief explanation of the structure of these formulae, that also justifies the name of the considered method, has as follows: Let $z \equiv E[g(X(0),X(1),\ldots,X(\tau_1))|M^*]$ be defined on the considered sample paths of the DTMC $\hat{\mathcal{M}}$, and τ_1 be the stopping time defined in Equation (37). Also, fix a policy $\bar{\zeta}_0$ in the underlying policy space, and let $P(\bar{\zeta}_0)$ denote the corresponding one-step transition probability for $\hat{\mathcal{M}}$. Finally, for any other policy $\bar{\zeta}$ and $t \in \mathbb{Z}_+$, define $L_t \equiv \prod_{i=1}^{t} P(\bar{\zeta})[X(i-1),X(i)]/P(\bar{\zeta}_0)[X(i-1),X(i)]$. Then, it can be checked that $z = \mathbb{E}_0[g(X(0),X(1),\ldots,X(\tau_1))L_{\tau_1}|M^*]$, where the notation $\mathbb{E}_0[\cdot]$ indicates that the expectation is taken with respect to the probability measure on the sample paths of $\hat{\mathcal{M}}$ that is defined by the policy $\bar{\zeta}_0$. Function L_{τ_1}, that facilitates this change of measure in the computation of z, is characterized as the corresponding *likelihood ratio*. Furthermore, it can be shown that under some regularity assumptions for the function $Z = g(X(0),X(1),\ldots,X(\tau_1))$, $\nabla\mathbb{E}[Z] = \nabla\mathbb{E}_0[ZL_{\tau_1}] = \mathbb{E}_0[Z\nabla L_{\tau_1}] = \mathbb{E}[Z\nabla L_{\tau_1}/L_{\tau_1}]$, where the gradient is taken with respect to $\bar{\zeta}$. Equation (41) is essentially the application of this last result to the sample-path functions of $\hat{\mathcal{M}}$ that define $z(\hat{r})$ and $z(\mathbf{1})$.

matrix P; as already established in Section 5.2.1, each element of P is a polynomial function of $\bar{\zeta}$, so the gradient exists.

It follows from the above discussion that an estimator of $\nabla\eta$ can be obtained from Equation (40) with the necessary estimates for $z(\mathbf{1})$, $\nabla z(\mathbf{1})$, $z(\hat{r})$ and $\nabla z(\hat{r})$ being obtained according to Equations (39) and (41). Let us denote those intermediate estimators by $\hat{z}(\omega, \mathbf{1})$, $\widehat{\nabla z}(\omega, \mathbf{1})$, $\hat{z}(\omega, \hat{r})$ and $\widehat{\nabla z}(\omega, \hat{r})$, where the employed sample path ω is included as an additional parameter of the estimators. Then, an estimator for the gradient can be derived as follows:

$$\widehat{\nabla\eta} = \frac{\hat{z}(\omega_1, 1) \cdot \widehat{\nabla z}(\omega_2, \hat{r}) - \hat{z}(\omega_3, \hat{r}) \cdot \widehat{\nabla z}(\omega_4, 1)}{\hat{z}(\omega_5, 1)\hat{z}(\omega_6, 1)} \tag{42}$$

However, the estimator of (42) is biased, in general, even if the six intermediate estimators in Equation (42) are obtained from independent sample paths.[4]

Fortunately, for the asymptotic-convergence result of Theorem 3, Conditions 12 and 13 in Section 5.1.1, which are related to the employed improvement directions Y_n, do not require the unbiasedness of the corresponding estimator for each iteration n of the algorithm; in particular, Condition 13 requires the unbiasedness of this estimator only as $n \to \infty$. This condition can be satisfied by establishing an "asymptotically unbiased" estimator (i.e., an estimator with its bias converging to zero as the sample size increases to infinity), and then setting the sample size as an increasing function of the iteration number n with no upper bounds. Furthermore, the direction defined by $\widehat{\nabla\eta}$ will be of an ascending nature if it is in acute angle with the vector $\nabla\eta$; but the angle between those two vectors is independent from the denominator of the right-hand-side of Equation (42). Hence, in the subsequent developments, we will (i) establish an estimator Y such that $\mathbb{E}[Y]$ is proportional to the numerator

[4]This bias is explained from the fact that for a function that is a ratio of two expectations, the corresponding estimator that takes the ratio between two respective estimators for the numerator and the denominator, will be biased [4]. The reader may observe this bias in the following example: There are two independent random variables X and Y with $\mathbb{P}\{X = 1\} = \mathbb{P}\{X = 2\} = \mathbb{P}\{Y = 1\} = \mathbb{P}\{Y = 2\} = 0.5$. Then, $\mathbb{E}[X]/\mathbb{E}[Y] = 1$, but $\mathbb{E}[X/Y] = 0.25 \times 0.5 + 0.5 \times 1 + 0.25 \times 2 = 1.125$!

part of the right-hand-side of Equation (40), i.e., $\mathbb{E}[Y] = C(z(\mathbf{1}) \cdot \nabla z(\hat{r}) - z(\hat{r}) \cdot \nabla z(\mathbf{1}))$ for some $C \in \mathbb{R}_+$; and (ii) adjust this estimator to an asymptotically unbiased estimator of $\nabla\eta$ by selecting appropriately the positive scalar C. We will also call the estimator established in the first step above as an *unbiased estimator of the improvement direction*.

An unbiased estimator of the improvement direction can be derived straightforwardly from the numerator part of the right-hand-side of Equation (40). More specifically, given two independent sample paths ω_1 and ω_2, the sought estimator can be built as

$$\tilde{Y} = \widehat{\nabla z}(\omega_1, \hat{r}) \cdot \hat{z}(\omega_2, \mathbf{1}) + \widehat{\nabla z}(\omega_2, \hat{r}) \cdot \hat{z}(\omega_1, \mathbf{1}) - \widehat{\nabla z}(\omega_2, \hat{r}) \cdot \hat{z}(\omega_1, \mathbf{1}) - \widehat{\nabla z}(\omega_1, \hat{r}) \cdot \hat{z}(\omega_2, \mathbf{1}) \tag{43}$$

Obviously, $\mathbb{E}[\tilde{Y}] = C\nabla\eta$ with $C = 2\mathbb{E}[\tau_1]^2$. Hence $\tilde{Y}$ is indeed an unbiased estimator for the improvement direction.

If N consecutive regenerative cycles are sampled and each regenerative cycle is treated as an independent sample path, then by grouping these sample paths into $\lfloor N/2 \rfloor$ pairs, we can construct $\lfloor N/2 \rfloor$ estimates of the improvement direction according to Equation (43), and eventually obtain the average of all these estimates:

$$\begin{aligned} Y_{\mathrm{A}} =& \frac{1}{\lfloor N/2 \rfloor} \sum_{j=1}^{\lfloor N/2 \rfloor} \Big(\widehat{\nabla z}(\omega_{2j-1}, \hat{r}) \cdot \hat{z}(\omega_{2j}, \mathbf{1}) + \widehat{\nabla z}(\omega_{2j}, \hat{r}) \cdot \hat{z}(\omega_{2j-1}, \mathbf{1}) \\ & - \hat{z}(\omega_{2j-1}, \hat{r}) \cdot \widehat{\nabla z}(\omega_{2j}, \mathbf{1}) - \hat{z}(\omega_{2j}, \hat{r}) \cdot \widehat{\nabla z}(\omega_{2j-1}, \mathbf{1}) \Big) \end{aligned} \tag{44}$$

Since $\mathbb{E}[Y_{\mathrm{A}}] = \mathbb{E}[\tilde{Y}] = 2\mathbb{E}[\tau_1]^2 \nabla\eta$, we propose to set

$$\begin{aligned} \widehat{\nabla\eta} =& \frac{Y_{\mathrm{A}}}{2\left(\frac{1}{N}\tau_N\right)^2} \\ =& \frac{N^2}{2\tau_N^2} Y_{\mathrm{A}} \end{aligned} \tag{45}$$

where τ_N is the stopping time that corresponds to the N-th revisit of the reference marking M^* (c.f. Equation (37)). Obviously, $\widehat{\nabla\eta}$ is an asymptotically unbiased estimator for the gradient $\nabla\eta$; i.e., as $N \to \infty$, $\mathbb{E}\left[\frac{N^2}{2\tau_N^2} Y_{\mathrm{A}}\right] \to \nabla\eta$.

It is also possible to use more than one regenerative cycle as a single sample path in the above computation. For instance, let each sample path in the estimator of Equation (43) contain K regenerative cycles, and divide the total budget of N regenerative cycles into $N/2K$ non-overlapping groups (to facilitate the following exposition, also suppose that $N/2K$ is an integer). Then, a single sample can be drawn from group i with mean $\mathbb{E}[\tilde{Y}_i^K] = K^2\mathbb{E}[\tilde{Y}] = 2K^2\mathbb{E}[\tau_1]^2\nabla\eta$:

$$\begin{aligned}\tilde{Y}_i^K = &\left(\sum_{j=K(i-1)}^{Ki} \widehat{\nabla z}(\omega_{2j-1}, \hat{r})\right)\left(\sum_{j=K(i-1)}^{Ki} \hat{z}(\omega_{2j}, \mathbf{1})\right) + \left(\sum_{j=K(i-1)}^{Ki} \widehat{\nabla z}(\omega_{2j}, \hat{r})\right) \\ &\left(\sum_{j=K(i-1)}^{Ki} \hat{z}(\omega_{2j-1}, \mathbf{1})\right) - \left(\sum_{j=K(i-1)}^{Ki} \hat{z}(\omega_{2j-1}, \hat{r})\right)\left(\sum_{j=K(i-1)}^{Ki} \widehat{\nabla z}(\omega_{2j}, \mathbf{1})\right) \\ &- \left(\sum_{j=K(i-1)}^{Ki} \hat{z}(\omega_{2j}, \hat{r})\right)\left(\sum_{j=K(i-1)}^{Ki} \widehat{\nabla z}(\omega_{2j-1}, \mathbf{1})\right) \end{aligned} \tag{46}$$

And the sample average obtained from all N cycles is

$$Y_{\mathrm{A}}^K = \frac{1}{N/2K}\sum_{i=1}^{N/2K} \tilde{Y}_i^K \tag{47}$$

with $\mathbb{E}[Y_{\mathrm{A}}^K] = \mathbb{E}[\tilde{Y}_i^K] = 2K^2\mathbb{E}[\tau_1]^2\nabla\eta$. Therefore, we have an estimator of $\nabla\eta$ as

$$\begin{aligned}\widehat{\nabla\eta} =& \frac{Y_{\mathrm{A}}^K}{2K^2\left(\frac{1}{N}\tau_N\right)^2} \\ =& \frac{N^2}{2K^2\tau_N^2} Y_{\mathrm{A}}^K \end{aligned} \tag{48}$$

As in the case of Equation (45), the estimator $\widehat{\nabla\eta}$ obtained from Equation (48) is asymptotically unbiased. In fact, (45) is a special case of (48) where $K = 1$.

For a given number of regenerative cycles N, it is an interesting problem to determine the number K of regenerative cycles contained in each sample path of Equation (46) so that the variance of the estimator in (48) is minimized. In general, a small value for K will provide more samples $\tilde{Y}_i^K$ for the estimator Y_{A}^K, and according to the Central Limit Theorem [71], will render the distribution of Y_{A}^K more similar to the Normal distribution, which is favorable in a statistical sense. On the other hand,

if K is too small, our empirical studies indicate that the variance of the resulting estimator $\widehat{\nabla\eta}$ will become larger for a fixed N.

Selecting the reference marking M^* Due to the employment of the randomization factor δ by the considered policy spaces, the Markov chain $\hat{\mathcal{M}}$ is irreducible, and any tangible marking, such as the first reachable tangible marking from the empty state, can be picked as the reference marking M^*. However, the markings with shorter recurrent times are preferred, since more samples can be drawn under a fixed budget of total transitions. Therefore, a good selection policy is to start the simulation from the empty state and run it for a certain number of transitions, recording the frequency of the visits of the generated sample path to each reached tangible marking. Then, the marking that has been visited most is picked as the reference marking M^*, since it is likely to have a shorter regenerative cycle.

The simulation running for the selection of M^* that was described in the previous paragraph will be called a *trial run* in the following. The reader should also notice that since the values of the decision variables $\bar{\zeta}$ change at each iteration, the recurrent times of the corresponding Markov chain $\hat{\mathcal{M}}$ will also be different. Thus, a new trial run should be performed at each iteration, and the reference markings at different iterations might be different.

Computing the one-step transition probabilities $P[X(k-1), X(k)]$ of the uniformized DTMC $\hat{\mathcal{M}}$ and their gradients It is evident from all the previous discussion in this section, that in order to compute the gradient estimators, or more essentially, the improvement directions for the SA recursion (31), we must be able to compute the intermediate estimators $\hat{z}(\omega, \mathbf{1})$, $\widehat{\nabla z}(\omega, \mathbf{1})$, $\hat{z}(\omega, \hat{r})$ and $\widehat{\nabla z}(\omega, \hat{r})$ that were defined in the previous paragraphs. $\hat{z}(\omega, \mathbf{1})$ can be obtained immediately from the counter of the DTMC transitions at the completion of a regenerative cycle. Also, $\hat{z}(\omega, \hat{r})$ is the accumulation of the immediate rewards of the visited states, $\hat{r}[X]$, during

such a cycle. But the computation for the LR gradient estimators $\widehat{\nabla z}(\omega, \mathbf{1})$ and $\widehat{\nabla z}(\omega, \hat{r})$ requires the computation of the ratio $\frac{\nabla P[X(k-1),X(k)]}{P[X(k-1),X(k)]}$ for the visited tangible markings at DTMC periods $k-1$ and k, and this is the topic to be discussed next.

We start by reminding the reader that the computation of the transition rate from a given tangible marking $M_{\mathcal{T}}$ to any other tangible marking $M'_{\mathcal{T}}$ has already been addressed in Section 2.3.3. After the application of the uniformization procedure with uniformizing rate r_{u} that was discussed in Section 3.3, we can also obtain the one-step transition probability from a tangible marking $M_{\mathcal{T}}$ to another tangible marking $M'_{\mathcal{T}}$ in the resulting DTMC $\hat{\mathcal{M}}$, as follows:

$$P[M_{\mathcal{T}}, M'_{\mathcal{T}}] = \sum_{t \in \mathcal{E}_t(M_{\mathcal{T}})} \frac{r(t)}{r_{\mathrm{u}}} \sum_{\sigma : \hat{M} = tr(M_{\mathcal{T}}, t) \wedge M'_{\mathcal{T}} = tr(\hat{M}, \sigma)} x_{\hat{M},\sigma} \tag{49}$$

In (49), σ denotes a (possibly empty) $\hat{\Pi}_2$-feasible firing sequence of untimed transitions leading from some marking $\hat{M}$ that results from the firing of an enabled transition t in marking $M_{\mathcal{T}}$, to the tangible marking $M'_{\mathcal{T}}$. And the quantity $x_{\hat{M},\sigma}$ is defined as the probability $\mathbb{P}\{\sigma \text{ is fired} | \hat{M}, \bar{\zeta}\}$. The detailed computation of $x_{\hat{M},\sigma}$, for any given pair of $\hat{M}$ and σ, is supported by Algorithm 1 at the end of Section 2.3.3.

Next we will focus on the computation of the vector $\nabla P[M_{\mathcal{T}}, M'_{\mathcal{T}}]$ for a given pair of tangible markings $M_{\mathcal{T}}$ and $M'_{\mathcal{T}}$ that constitute consecutive tangible markings in the simulation sample paths that are employed by the considered estimator. Applying the gradient operator with respect to $\bar{\zeta}$ on Equation (49), we have:

$$\begin{aligned} \nabla P[M_{\mathcal{T}}, M'_{\mathcal{T}}] &= \nabla \left(\sum_{t \in \mathcal{E}_t(M_{\mathcal{T}})} \frac{r(t)}{r_{\mathrm{u}}} \sum_{\sigma : \hat{M} = tr(M_{\mathcal{T}}, t) \wedge M'_{\mathcal{T}} = tr(\hat{M}, \sigma)} x_{\hat{M},\sigma} \right) \\ &= \sum_{t \in \mathcal{E}_t(M_{\mathcal{T}})} \frac{r(t)}{r_{\mathrm{u}}} \sum_{\sigma : \hat{M} = tr(M_{\mathcal{T}}, t) \wedge M'_{\mathcal{T}} = tr(\hat{M}, \sigma)} \nabla x_{\hat{M},\sigma} \end{aligned} \tag{50}$$

The second equation above applies since neither the firing rates of timed transitions nor the uniformization rate depends on $\bar{\zeta}$. From Equation (50) we can see that the remaining task is the calculation of the gradient of $x_{\hat{M},\sigma}$.

Suppose that $\sigma = \hat{t}_1 \ldots \hat{t}_m$ and $m \geq 0$. First define a sequence of markings $\hat{M}_i, i = 1, \ldots, m$, where $\hat{M}_1 \equiv \hat{M}$ and $\hat{M}_i \equiv tr(\hat{M}_{i-1}, \hat{t}_{i-1})$ for $i \geq 2$. Then, according to Proposition 3 in Section 3.1.2, these markings are different from each other. Next, define two sets of transition probabilities $\tilde{p}_i$ and $\hat{p}_i$, for $i = 1, \ldots, m$, where:

$$\tilde{p}_i \equiv \mathbb{P}\{\hat{t}_1 \ldots \hat{t}_i \text{ is fired} | \hat{M}_1, \bar{\zeta}\}$$

$$\hat{p}_i \equiv \mathbb{P}\{\hat{t}_i \text{ is fired} | \hat{M}_i, \bar{\zeta}\}$$

For $i = 0$, we also set $\tilde{p}_0 = \hat{p}_0 = 1$. Obviously, $x_{\hat{M},\sigma} = \tilde{p}_m = \prod_{i=0}^{m} \hat{p}_i$. Also, the probability $\tilde{p}_i$ can be expressed in an inductive way:

$$\tilde{p}_i = \tilde{p}_{i-1}\hat{p}_i, i = 1, \ldots, m \tag{51}$$

with the initial condition $\tilde{p}_0 = \hat{p}_0 = 1$.

Next, let us focus on the partial derivative $\frac{\partial x_{\hat{M},\sigma}}{\partial \bar{\zeta}[\hat{k}]} = \frac{\partial \tilde{p}_m}{\partial \bar{\zeta}[\hat{k}]}$, where $\bar{\zeta}[\hat{k}]$ is some specific component of $\bar{\zeta}$. Then, from Equation (51), the corresponding partial derivative on $\tilde{p}_i$ can be obtained through the following recursion:

$$\frac{\partial \tilde{p}_i}{\partial \bar{\zeta}[\hat{k}]} = \frac{\partial \tilde{p}_{i-1}}{\partial \bar{\zeta}[\hat{k}]}\hat{p}_i + \tilde{p}_{i-1}\frac{\partial \hat{p}_i}{\partial \bar{\zeta}[\hat{k}]}, i = 1, \ldots, m \tag{52}$$

with the initial condition $\tilde{p}_0 = 1$ and $\frac{\partial \tilde{p}_0}{\partial \bar{\zeta}[\hat{k}]} = 0$.

More specifically, the items $\tilde{p}_i$ and $\frac{\partial \tilde{p}_i}{\partial \bar{\zeta}[\hat{k}]}$ in the above equation can be obtained recursively through Equations (51) and (52). On the other hand, for the computation of $\hat{p}_i$ and $\frac{\partial \hat{p}_i}{\partial \bar{\zeta}[\hat{k}]}$, we notice the following: If $|\mathcal{E}_u^{\hat{\Pi}_2}(\hat{M}_i)| = 1$, then the transition $\hat{t}_i$ corresponds to the only available option of a "degenerate" random switch, and $\hat{p}_i = 1, \frac{\partial \hat{p}_i}{\partial \bar{\zeta}[\hat{k}]} = 0$. Otherwise, $\hat{p}_i$ is determined by the corresponding random switch, i.e., $\hat{p}_i = Z_{\hat{M}_i}[\hat{t}_i]$, and $\frac{\partial \hat{p}_i}{\partial \bar{\zeta}[\hat{k}]}$ is determined according to the expression of $Z_{\hat{M}_i}[\hat{t}_i]$ in the adopted policy space.

In the following discussion we assume that the considered random switch is a static or dynamic random switch that is defined directly by the corresponding transition selection probabilities. The case of random switches of the partially disaggregated

type will be discussed in Section 5.4 at the end of this chapter, that addresses the development of the corresponding SA algorithm. Then, for the considered set of random switches, the probability $\hat{p}_i$ takes the form:

$$\hat{p}_i = \begin{cases} \bar{\zeta}[idx(\hat{M}_i, \hat{t}_i)] & \text{if } idx(\hat{M}_i, \hat{t}_i) \neq \texttt{NULL} \\ 1 - \sum_{t:idx(\hat{M}_i,t)\neq\texttt{NULL}} \bar{\zeta}[idx(\hat{M}_i, t)] & \text{otherwise} \end{cases} \tag{53}$$

From the structure of the right-hand-side of Equation (53) we can infer that

$$\frac{\partial \hat{p}_i}{\partial \bar{\zeta}[\hat{k}]} = \begin{cases} 1 & \text{if } idx(\hat{M}_i, \hat{t}_i) = \hat{k} \\ -1 & \text{if } idx(\hat{M}_i, \hat{t}_i) = \texttt{NULL} \\ 0 & \text{otherwise} \end{cases} \tag{54}$$

The first branch in Equation (54) corresponds to the case where the untimed transition associated to $\bar{\zeta}[\hat{k}]$ is the fired transition $\hat{t}_i$; hence, the corresponding selection probability $\hat{p}_i$ is characterized by the first branch in Equation (53) and the corresponding partial derivative is equal to 1. The second branch in Equation (54) corresponds to the case where the untimed transition associated to $\bar{\zeta}[\hat{k}]$ is different from the fired transition $\hat{t}_i$, but it appears in the specification of the corresponding selection probability $\hat{p}_i$ which is characterized by the second branch in Equation (53); hence, the corresponding partial derivative is equal to -1. Finally, the third branch in Equation (54) covers all the remaining cases. More specifically, in these cases, the untimed transition associated to $\bar{\zeta}[\hat{k}]$ is not involved in the computation of $\hat{p}_i$; hence, the corresponding partial derivative is equal to 0.

Remark When the selection probabilities of all random switches are represented by Equation (53), as is the case that was discussed above, then, it is obvious that $P[M_{\mathcal{T}}, M'_{\mathcal{T}}]$ is a polynomial function of the decision variables $\bar{\zeta}$. Therefore, in this case $\nabla P[M_{\mathcal{T}}, M'_{\mathcal{T}}]$ can also be expressed in closed-form (c.f. [59]). However, if the underlying problem formulation employs also partially disaggregated random switches according to the representation forms that were discussed in Section 4.3, then the

Algorithm 4 Implementing a single (possibly self-loop) transition from one tangible marking to another tangible marking in the simulation of the DTMC $\hat{\mathcal{M}}$ that is employed by the proposed SA algorithm

Input: $GSPN^S$, Π, r_{u}, ξ, $M_{\mathcal{T}}$, Ξ, $\bar{\zeta}$, $idx(\cdot)$.
Output: $M'_{\mathcal{T}}$, $P[M_{\mathcal{T}}, M'_{\mathcal{T}}]$, $\nabla P[M_{\mathcal{T}}, M'_{\mathcal{T}}]$, Ξ, $\bar{\zeta}$, $idx(\cdot)$.
1: $dest \leftarrow \{M_{\mathcal{T}}\}$; $\tilde{p}[M_{\mathcal{T}}] \leftarrow 1 - \frac{1}{r_{\mathrm{u}}} \sum_{t \in \mathcal{E}_t(M_{\mathcal{T}})} r(t)$; $\nabla \tilde{p}[M_{\mathcal{T}}] \leftarrow \mathbf{0}$.
2: **for each** $t \in \mathcal{E}_t(M_{\mathcal{T}})$ **do**
3: $\quad \hat{M} \leftarrow tr(M_{\mathcal{T}}, t)$.
4: $\quad$ Get *pair* from modified Algorithm 1, also update Ξ, $\bar{\zeta}$.
5: $\quad$ **for each** $(\sigma, x) \in pair$ **do**
6: $\quad\quad M'_{\mathcal{T}} \leftarrow tr(\hat{M}, \sigma)$.
7: $\quad\quad$ **if** $M'_{\mathcal{T}} \notin dest$ **then**
8: $\quad\quad\quad dest \leftarrow dest \cup \{M'_{\mathcal{T}}\}$; Extend $\tilde{p}$ and $\nabla \tilde{p}$ with zero components.
9: $\quad\quad$ **end if**
10: $\quad\quad$ Compute ∇x according to Eqs. (51)–(54).
11: $\quad\quad \tilde{p}[M'_{\mathcal{T}}] \leftarrow \tilde{p}[M'_{\mathcal{T}}] + \frac{r(t)}{r_{\mathrm{u}}} x$; $\nabla \tilde{p}[M'_{\mathcal{T}}] \leftarrow \nabla \tilde{p}[M'_{\mathcal{T}}] + \frac{r(t)}{r_{\mathrm{u}}} \nabla x$;
12: $\quad$ **end for**
13: **end for**
14: Select $M'_{\mathcal{T}}$ according to probability defined in $\tilde{p}[\cdot]$.
15: **return** $M'_{\mathcal{T}}$, $\tilde{p}[M'_{\mathcal{T}}]$, $\nabla \tilde{p}[M'_{\mathcal{T}}]$, Ξ, $\bar{\zeta}$, $idx(\cdot)$.

expression of $P[M_{\mathcal{T}}, M'_{\mathcal{T}}]$ is no longer a polynomial function. And things become more complicated if the same enabling pattern for the partially disaggregated random switches appears more than once along the sequence σ.[5] From this perspective, the computation of $P[M_{\mathcal{T}}, M'_{\mathcal{T}}]$ and $\nabla P[M_{\mathcal{T}}, M'_{\mathcal{T}}]$ based on the recursive schemes that are defined by Equations (51) and (52), are more convenient in numerical implementation.

Algorithm 4 supports the implementation of a single (possibly self-loop) transition from one tangible marking to another tangible marking in the employed simulations of the DTMC $\hat{\mathcal{M}}$, returning also the corresponding one-step transition probability and its gradient. This algorithm takes as input the following parameters that are extracted from the tuple $\langle \mathcal{P}, \mathcal{T}_u, \mathcal{T}_t, \mathcal{F}, \mathcal{W}, M_0, r, \hat{r} \rangle$, the augmentation of the corresponding GSPN model with the relevant reward functions: the GSPN structure $GSPN^S = \langle \mathcal{P}, \mathcal{T}_u, \mathcal{T}_t, \mathcal{F}, \mathcal{W}, r \rangle$; the constraints imposed by the policy space Π (e.g.,

[5]The reader should notice that even though the markings M_i that are visited by the considered sequence σ are guaranteed to be different, it is still possible that the sets of the enabled transitions in these markings are the same.

the DAP, the non-deliberately-idling requirement, the refinement, etc.), represented by the application of the Π-conditional notation; the uniformization rate r_{u}. Besides the above, the input to this algorithm also includes: a mechanism denoted as ξ that initializes the new random switches that are discovered during the enumeration of the corresponding local subspace of vanishing markings; the current tangible marking $M_{\mathcal{T}}$; the current set of all the random switches Ξ and their corresponding decision variables coded in a vector $\bar{\zeta}$ according to the index-searching function $idx(\cdot)$.

The output obtained from the algorithm includes: the destination tangible marking $M'_{\mathcal{T}}$; the updated set of Ξ, the extended vector $\bar{\zeta}$ and the extended function $idx(\cdot)$, if new random switches are discovered during the implementation of the algorithm; the transition probability $P[M_{\mathcal{T}}, M'_{\mathcal{T}}]$ under the current policy $\bar{\zeta}$ and the gradient $\nabla P[M_{\mathcal{T}}, M'_{\mathcal{T}}]$ with respect to $\bar{\zeta}$. If static random switches are applied, then the update can be on $idx^S(\cdot)$ instead of $idx(\cdot)$.

The variables $\tilde{p}$ and $\nabla\tilde{p}$ in Algorithm 4 are labeled arrays with dynamic lengths, and their lengths are always equal to the size of the set *dest* that is defined in Algorithm 4 and holds all the tangible markings $M'_{\mathcal{T}}$ that have been recognized as possible successors of $M_{\mathcal{T}}$. Each element in $\tilde{p}$ is a scalar, recording the transition probability from $M_{\mathcal{T}}$ to the corresponding destination tangible marking in the set *dest*; Each element in the array $\nabla\tilde{p}$ is a vector, recording the gradient of the corresponding transition probability.[6]

Validation of Conditions 12 and 13 in Section 5.1.1 As already mentioned, the estimator $\widehat{\nabla\eta}$ obtained from Equation (48) (or Equation (45)) is asymptotically

[6] Also, Algorithm 1 must be slightly modified before it can be called in Algorithm 4. First, the input random switches include the information of the random switches already defined in Ξ with $\bar{\zeta}$ and $idx(\cdot)$, and the initialization mechanism ξ; besides the GSPN structure and the current marking M, the input also includes the constraints introduced from policy space Π; the output includes the updates on the random switches besides the set of pairs. Next, the Π-conditional sets $\mathcal{E}_u^{\Pi}(M)$ and $\mathcal{R}_{\mathcal{T}}^{\Pi}$ replace the original sets $\mathcal{E}_u(M)$ and $\mathcal{R}_{\mathcal{T}}$. Finally, at Line 8, an extra operation is performed for the newly discovered random switches: update the set Ξ, extend the vector $\bar{\zeta}$ and the function $idx(\cdot)$, and initialize the new components of $\bar{\zeta}$.

unbiased. In the developed SA algorithm, the employed improvement direction Y_n will be the estimator $\widehat{\nabla\eta}$ that is obtained from Equation (48). Also, the employed sample size $N/2K$ will be a function $g(n)$ of the iteration number n with $\lim_{n\to\infty} g(n) = \infty$ (notice that the growth of this function can be sublinear with respect to n; for instance, we can set $g(n) = \sqrt{n}$). Then, $\lim_{n\to\infty} \mathbb{E}[Y_n] = \nabla\eta(\bar{\zeta}_n)$, and the next lemma results immediately from the previous remarks.

Lemma 5 *Consider the improvement direction Y_n obtained by Equation (48) by setting $N = 2K \cdot g(n)$, where $g(n)$ is a function of the iteration number n in the SA recursion (31), and further assume that $\lim_{n\to\infty} g(n) = \infty$. Then, the sequence $\{Y_n : n \in \mathbb{Z}_+\}$ satisfies Condition 13 in Section 5.1.1.*

Furthermore, from the previous developments regarding the calculation of the estimator $\widehat{\nabla\eta}$ in the context of the MP formulation (33)–(35), it is obvious that the variance of $\widehat{\nabla\eta}$ is finite. Therefore, the next lemma also holds.

Lemma 6 *The improvement direction defined by Equation (48) satisfies Condition 12 in Section 5.1.1.*

It remains to specify the step size schedule for the SA recursion (31) in a way that satisfies Condition 11 of Section 5.1.1, and the implementation of the projection operator $\texttt{proj}_H(\cdot)$. These two topics will be addressed in the next section.

5.2.3 Specification of the remaining ingredients of the SA recursion

There are a number of alternative schedules for the step size γ_n that satisfy Condition 11 in Section 5.1.1. Some of the most commonly used are those defined by the sequences $\frac{1}{n+1}$ and $\frac{a}{n+b}$, where the positive scalar parameters a and b can adjust respectively the amplitude and the speed for the decrease of the step size γ_n. In the subsequent developments, we shall use the function $\gamma(n)$ to denote any function that returns the step size at iteration n and satisfies Condition 11 in Section 5.1.1.

The implementation of the projection operator $\texttt{proj}_H(\cdot)$ with respect to the polyhedron that is defined by the constraints (34) and (35) is a well addressed topic in the existing literature. An algorithm that implements this projection is provided in Appendix D for completeness. This algorithm is derived from the standard MP-based characterization of the notion of projection, and fairly standard arguments and techniques coming from the area of nonlinear programming.

5.2.4 An asymptotically convergent SA algorithm for the considered optimization problem

By this point, we have detailed all the key ingredients that are involved in the implementation of the basic SA algorithm of Section 5.1, and we have also established that the specification of all these ingredients abides to Conditions 8–13 that are required by Theorem 3 that establishes the asymptotic convergence of that algorithm. Next, we provide a detailed articulation of the adaptation of the basic SA algorithm to the considered problem context, and establish formally its asymptotic convergence.

The considered algorithmic implementation is presented in Algorithm 5. The input to this algorithm includes a RAS instance Φ, the randomization factor δ, the random switch initialization mechanism ξ, and some additional algorithmic parameters that include: the parameter *trial* which is the number of transitions used to identify the reference marking M^* at each iteration; the function $\gamma(\cdot)$ that determines the step size at each iteration; the function $g(\cdot)$ that determines the extent of sampling to be employed in the various simulation runs; the parameter K that is employed in the gradient estimation according to Equation (48); and the parameter $n_{\max}$ that specifies the total number of iterations. We also notice that since the sample size $g(n)$ for computing the improvement direction $\widehat{\nabla\eta}$ essentially specifies the number of the $\tilde{Y}$ samples, i.e., the value of $N/2K$ in the discussion of Section 5.2.2, the number of the sampled regenerative cycles N in Equation (48) should be set equal to $2K \cdot g(n)$.

Algorithm 5 An implementation of the basic stochastic approximation algorithm for the considered optimization problem

Input: RAS Φ, δ, ξ, $trial$, $\gamma(\cdot)$, $g(\cdot)$, K, $n_{\max}$.
Output: GSPN, DAP, Ξ, $\bar{\zeta}$, $idx^S(\cdot)$.

Model RAS Φ as GSPN with the methods in Section 3.1.
Solve for the DAP of Φ with the methods in Section 2.2 and code the DAP and the non-deliberately-idling constraint into GSPN as Π_2. The finally adopted policy space should be $\hat{\Pi}_3^S$.
for $n = 1 \to n_{\max}$ **do**
  Starting from the initial marking M_0, simulate the GSPN for $trial$ non-self-loop transitions to get the most visited tangible marking M^*. Also extend Ξ, $\bar{\zeta}$ and $idx^S(\cdot)$ with any newly encountered random switches, and initialize the new components of $\bar{\zeta}$ according to ξ.
  $t \leftarrow 0$; $x \leftarrow$ `odd`.
  for $i = 1 \to g(n)$ **do**
    Set the 8 counters for the respective 8 sums in Eq. (46) to 0 or $\mathbf{0}$; Set the single samples $\hat{z}(\omega, \mathbf{1})$, $\widehat{\nabla z}(\omega, \mathbf{1})$, $\hat{z}(\omega, \hat{r})$ and $\widehat{\nabla z}(\omega, \hat{r})$ to 0 or $\mathbf{0}$; $pair \leftarrow 0$.
    while $pair < K$ **do**
      $t \leftarrow t + 1$.
      Evolve the GSPN to its next tangible marking M' according to Algorithm 4, and update the single samples $\hat{z}(\omega, \mathbf{1})$, $\widehat{\nabla z}(\omega, \mathbf{1})$, $\hat{z}(\omega, \hat{r})$ and $\widehat{\nabla z}(\omega, \hat{r})$ according to Eqs. (39) and (41).
      if $M' = \bar{M}$ **then**
        Depending on whether x is labelled "odd" or "even", update the 4 out of 8 sums of Eq. (46) with odd or even subscripts.
        Set the single samples $\hat{z}(\omega, \mathbf{1})$, $\widehat{\nabla z}(\omega, \mathbf{1})$, $\hat{z}(\omega, \hat{r})$ and $\widehat{\nabla z}(\omega, \hat{r})$ to 0 or $\mathbf{0}$.
        if $x =$ `odd` **then**
          $x \leftarrow$ `even`.
        else
          $x \leftarrow$ `odd`; $pair \leftarrow pair + 1$.
        end if
      end if
    end while
    Compute $\tilde{Y}_i$ using Eq. (46).
  end for
  Compute Y_{A} using Eq. (47), then $\widehat{\nabla \eta}$ using to Eq. (48) with $N = 2K \cdot g(n)$ and $\tau_N = t$.
  $\bar{\zeta} \leftarrow \bar{\zeta} + \gamma(n) \cdot \widehat{\nabla \eta}$.
  Project $\bar{\zeta}$ with respect to every random switch onto the feasible region defined by Eqs. (34) and (35), using Algorithm 7 in Appendix D.
end for
return GSPN, DAP, Ξ, $\bar{\zeta}$, $idx^S(\cdot)$.

As for the computation that is performed by Algorithm 5, it can be briefly described as follows: The algorithm starts with a "pre-processing" phase that includes (i) the conversion of the input RAS performance optimization problem to a GSPN reward model, (ii) the computation and the deployment of a correct DAP for the obtained GSPN model, and (iii) an initialization process for the underlying policy space that (a) sets the vector of the decision variables $\bar{\zeta}$ to a zero-dimensional vector, (b) sets the set containing the identified static random switches Ξ equal to the empty set, and also (c) sets the index-searching function $idx^{S}(\cdot)$ to a "null" mapping, since at this point the domain of this function is the empty set.

Then, the "main" phase of Algorithm 5 consists of $n_{\max}$ iterations of the SA recursion of Equation (31), that seek to obtain an optimized pricing of the (so far discovered) static random switches of the underlying GSPN model that was generated in the pre-processing phase. At a typical SA iteration n, a "trial" simulation is initially implemented to determine the reference marking M^{*} to be used in the evaluation of the current policy $\bar{\zeta}_{n}$ with respect to the applying objective function $\eta(\bar{\zeta})$ and its gradient. Subsequently, another simulation samples $(2K \cdot g(n))$ regenerative cycles with respect to the selected reference marking M^{*} in order to estimate the gradient of the objective function $\eta(\bar{\zeta})$ with respect to the decision variables $\bar{\zeta}$; more specifically, the aforementioned simulation and the processing of its output are performed in Lines 6–23 of Algorithm 5 and they culminate in the estimates Y_{A} and $\widehat{\nabla\eta}$ of the improvement direction and the gradient of the objective function $\eta(\bar{\zeta})$, according to the logic that was presented in Section 5.2.2. Finally, the estimated gradient $\widehat{\nabla\eta}$ is employed in the SA recursion of Equation (31) in order to generate the policy $\bar{\zeta}_{n+1}$, to be considered in the next iteration.

During the aforementioned simulations that are employed in the main phase of Algorithm 5, every time that a vanishing marking M with a new enabling pattern $\mathcal{E}_{u}(M)$ is visited, the algorithm also updates the data structures that encode $\bar{\zeta}$, Ξ

and $idx^S(\cdot)$, and initializes the corresponding static random switch according to the prespecified mechanism ξ (c.f. Lines 4 and 10 of Algorithm 5 and Footnote 6 in the description of Algorithm 4).

The output returned by Algorithm 5 includes the GSPN (reward) model, the applied DAP for the GSPN, the set of the static random switches Ξ encountered by the algorithm, the vector $\bar{\zeta}$ that defines the randomized policy that is computed by the algorithm, and the corresponding index-searching function $idx^S(\cdot)$.

The next theorem provides a formal statement for the asymptotic convergence of Algorithm 5. A formal proof of this theorem can be based on Theorem 3, Lemmas 2–6, and the further specification of the function $\gamma(\cdot)$ in Section 5.2.3.

Theorem 4 *Consider the implementation of Algorithm 5 on a given instance Φ from the RAS class that was defined in Section 2.1. Then, as $n \to \infty$, the algorithm iterate $\bar{\zeta}_n$ will converge with probability 1 to a compact and connected set of points satisfying Condition 7 for the corresponding MP formulation of Equations* (33)–(35).

5.2.5 Some additional considerations

Local vs. global optimality Another issue regarding Algorithm 5 that needs some further attention, pertains to the fact that the (asymptotic) results of this algorithm are local optima for the MP formulation of Equations (33)–(35). This is a general limitation for all gradient-based algorithms for non-convex nonlinear optimization problems. A typical remedy to this problem is the repetitive execution of this algorithm while starting from many different points in the feasible region H.

In particular, if the set S_0 of these starting points is selected from the solution space H randomly according to a uniform distribution, and the number of the algorithm executions is sufficiently large, then there are considerable chances that the estimated best solution in S_0 is also a good solution with respect to the whole solution space H. More formally, for any point $\bar{\zeta} \in S_0$, let $\bar{\zeta}^*$ denote the (asymptotic)

solution that is returned by Algorithm 5 when initialized at $\bar{\zeta}$. Then, if each point in set S_0 is picked from H through a uniform distribution, the point $\bar{\zeta}^*_{S_0}$ corresponding to $\bar{\zeta}_{S_0} \equiv \arg\max_{\bar{\zeta}\in S_0}\eta(\bar{\zeta}^*)$ will be among the top $100\beta\,\%$ solutions in H (i.e., $\mathbb{P}\{\eta(\bar{\zeta}^*_{S_0}) > \eta(\bar{\zeta}) \mid \bar{\zeta} \text{ is uniformly picked from } H\} > 1-\beta$) with a certain confidence $100(1-\alpha)\%$, by choosing appropriately the cardinality of S_0. Problems of this nature are currently addressed by an area that is called *ordinal optimization* [43], and it can be seen as a pre-processing stage to be executed before the initiation of the so-called *cardinal optimization* stage, which involves the (repetitive execution of the) SA algorithm that has been developed for our problem.

Some alternative SA algorithms Algorithm 5 adopts the rather typical approach where, at any iteration n, the decision variables $\bar{\zeta}$ are kept fixed to their current values $\bar{\zeta}_n$ during the simulations of the DTMC $\hat{\mathcal{M}}$ that are necessary for the generation of the improvement direction Y_n. The literature also avails of some more aggressive updating schemes, where the decision variables $\bar{\zeta}$ are updated more frequently and simultaneously with the data that provide the employed improvement directions. Such SA schemes can be found in [68, 51, 28], where they are also proven to be asymptotically convergent for problem structures and conditions similar to those that apply to the considered optimization problem. Furthermore, the policy updating schemes that are employed by these algorithms render them much less intrusive to the operation of the underlying system, and therefore, they can be implementable in an "on-line" mode, effecting a "learning capability" for the overall operation. But those algorithms often present a very large fluctuation in the specification of the applied policies due to the large variance of the improvement direction estimator [68], or they require the availability of additional information for the underlying value functions for a more stable execution [51, 28]. Our own experience with the testing of these algorithms on the problem that is considered in this work confirms the above remarks.

Hence, we have decided to defer the potential implementation of these algorithms to the considered problem as a topic for future research.

5.3 A more practical implementation of the SA algorithm

Theorem 4 in the previous section established the asymptotic convergence of Algorithm 5. However, no information can be retrieved from that result on the *transient* behavior of the algorithm. Yet, this behavior is really important when SA algorithms are considered from a more practical standpoint. In any practical implementation of these algorithms, the key requirement is the ability of the algorithm to move in a reliable and expedient manner towards policies that have a better performance than the running policy and can be reached from the current solution in the stepwise manner that is adopted by this class of algorithms. This reliability and expediency are especially important when the algorithm is "climbing its way" towards the corresponding locally optimal solution, while it is also recognized that once the algorithm gets to a solution that exhibits a performance pretty comparable to that of the considered local optimum, further progress might be difficult due to the flatness of the response surface in that neighborhood. In the following, we shall refer to such flat areas around a local optimum as a "plateau". Then, making use of this concept, we can summarize the above discussion about the practical considerations for our algorithm as *a requirement that the algorithm will be able to reach a plateau of the response surface that is accessible from its starting solution* $\bar{\zeta}_0$*, in a reliable and expedient manner. Furthermore, in order to avoid the waste of valuable computational resources, we also require that the algorithm should be able to detect its access of a plateau in a timely manner, and terminate its exploration at that point.*

In this section, we present a modified version of Algorithm 5 that seeks to satisfy the aforestated requirements. More specifically, we seek to meet these requirements by making the following changes and additions to the algorithm: (i) First, we replace

the original step-size schedules $\gamma(n)$, that were discussed in Section 5.2.3, with a small but constant step size γ, since past research has established that such small but constant step sizes can guarantee the progress of the algorithm towards a local optimum in a more reliable manner [9]. (ii) We also introduce additional methodology based on the theory of sampling and statistical inference that will help us determine the amount of sampling to be performed in the computation of the improvement direction in a way that controls the variability in the outcome of this computation. (iii) Finally, we also employ statistical inference in combination with Condition 7 of Section 5.1 in order to develop a statistical test for detecting the algorithm access to one of the aforementioned plateaus.

We start the presentation of these developments by introducing in the next section a first variation of Algorithm 5 that operates with a constant step size and also a constant sample size for the estimation of the improvement direction. The presentation of this algorithm and some ensuing experimentation with it intends to (i) reveal the practical advantages of the adoption of a constant step size, and (ii) function as a "baseline" for the developments of the subsequent sections that resolve the determination of the necessary sampling and the design of the employed termination condition through the more sophisticated statistical analysis that was mentioned above.

5.3.1 A "baseline" experiment

In this section we perform a numerical experiment with an adaptation of Algorithm 5 that employs a constant step size and also a constant sample size for the determination of the improvement direction. In the sequel, this adapted version of Algorithm 5 will be called the"baseline algorithm". The baseline algorithm and the aforementioned experiment are meant to provide a benchmark for the subsequent developments that were outlined in the opening part of Section 5.3. Although the baseline algorithm is not convergent in the asymptotic sense, our empirical study will show that it is

still able to lead to improved solutions, and eventually have the returned solution wandering at some near-optimal region.

In the considered experiment, the baseline algorithm is implemented on the first 16 smaller CRL configurations of Table 4.1, under the further assumption that the rates of all the timing distributions involved are set equal to one. Furthermore, the adopted policy space is $\hat{\Pi}_3^S$, with the employed DAP being the maximally permissive DAP for each CRL, and the corresponding refinement of the original policy space Π_2 being performed by means of Algorithm 3 in Section 4.1.4. We fix the algorithm step size to $\gamma(\cdot) \equiv 2$, and we also set the number of the sampled regenerative cycles, N, (c.f. Equation (48) in Section 5.2.2) to $N = 10,000$, for all SA iterations. The initial solution $\bar{\zeta}_0$ is a "totally random" policy that equalizes the firing probabilities for all the $\hat{\Pi}_2$-enabled untimed transitions in every static random switch that is encountered by the algorithm. Finally, we also impose a randomization factor $\delta = 0.1$, and we further set $trial = 3,000$, $K = 100$, and $n_{\max} = 500$.

The 16 CRL configurations selected for this experiment are still small enough to have their steady-state average reward for any given policy $\bar{\zeta}$ evaluated through value iteration [82], to any desired precision. Table 5.1 reports the long-run average throughput η, evaluated through value iteration, at the solutions $\bar{\zeta}_0$, $\bar{\zeta}_{100}$ and $\bar{\zeta}_{500}$ returned by the baseline algorithm at the corresponding iterations 0, 100 and 500, with a precision of 5 digits after the decimal point. Particularly, the value at $\bar{\zeta}_0$ is the objective value at the initial solution, i.e., the CRL throughput under the totally random policy; on the other hand, the value at $\bar{\zeta}_{500}$ is the true value of the objective function at the solution that is returned by the baseline algorithm upon its completion.[7] Furthermore, the second column of Table 5.1, that is entitled "Max",

[7]Obviously, in the application of the considered SA algorithms on more general CRL / RAS configurations, the evaluation of the returned solution through value iteration will not be practically possible, since the underlying state space may become intractable, and we must resort to simulation for the support of this task.

Table 5.1: The values of the objective function at the solutions obtained in the baseline experiment for the first 16 CRL configurations of Table 4.1

Conf.	Objective values				Trans.	Time
	Max	$\eta(\bar{\zeta}_0)$	$\eta(\bar{\zeta}_{100})$	$\eta(\bar{\zeta}_{500})$	($\times 10^6$)	(sec)
1	0.48000	0.47333	0.47991	0.47995	28.53	56
2	0.44444	0.43478	0.44324	0.44324	23.29	42
3	0.49254	0.48495	0.48927	0.48926	70.84	191
4	0.49959	0.49811	0.49865	0.49874	152.27	369
5	0.50000	0.50000	0.50000	0.50000	283.28	680
6	0.46411	0.45250	0.46195	0.46208	53.45	99
7	0.49310	0.48348	0.48565	0.48664	258.97	530
8	0.49820	0.49225	0.49462	0.49539	291.03	642
9	0.49999	0.49969	0.49966	0.49984	544.38	1,282
10	0.32234	0.30649	0.30825	0.31180	2863.49	8,186
11	0.43734	0.43359	0.43514	0.43535	1213.33	3,386
12	0.42225	0.41539	0.42074	0.42113	356.46	1,118
13	0.43212	0.41808	0.42351	0.42422	684.71	1,889
14	0.41063	0.39231	0.40095	0.40291	263.97	673
15	0.37667	0.37359	0.37448	0.37536	2813.46	9,210
16	0.35729	0.35453	0.35549	0.35655	1481.54	4,405

reports the maximum value of the objective function η that is obtained through the solution of the MDP that is developed in Appendix A; these values can be considered as an upper bound of η that is attained in the policy space Π_2.[8] The last two columns report the number of Markovian transitions and the running time in seconds for the simulation of the total 500 iterations for each CRL configuration. The computing time for each configuration is evaluated on a Windows 7 PC with Intel CPU E5-2680 v3, 2.5 GHz and 6 GB of memory (RAM).

[8] We remind the reader that the refinement that defines the policy space $\hat{\Pi}_2$ from the original policy space Π_2 maintains the performance potential of the latter, but, in general, the policy space $\hat{\Pi}_3^S$ does not maintain the performance potential of $\hat{\Pi}_2$. This compromise of the performance potential is due to (i) the employment of the randomization factor δ that distinguishes $\hat{\Pi}_3$ from $\hat{\Pi}_2$, and (ii) the coupling effects among the decision variables that are introduced by the static random switches, as we move from the policy space $\hat{\Pi}_3$ to $\hat{\Pi}_3^S$. The performance degradation due to the randomization factor δ is expected to be rather minor as long as the value of δ is kept at pretty small levels. On the other hand, the performance degradation due to the introduction of static random switches can be addressed through the partial disaggregation that was discussed in Section 4.3 and it is further revisited in Section 5.4 at the end of this chapter.

We can observe from Table 5.1 that even with such a straightforward implementation of the considered SA algorithm, we can attain significant improvement against the initial solution, except for configurations #5 and #9 where the "performance gaps" between the corresponding initial solutions and the maxima obtained from the MDP solutions are too small. On the other hand, we should also point out, for completeness, that the values reported in column "Max" might not be attainable by the considered algorithm due to (i) the reasons discussed in Footnote 8, and (ii) the constant step size that is employed by the algorithm and will prevent it from settling to the locally optimal solution.

A careful observation of the data reported in Table 5.1 gives rise to the following two issues regarding the behavior of the baseline algorithm. The first of these issues has to do with the fact that some configurations do not need as many as 500 iterations for the computation of a good scheduling policy, since almost no improvement is made during the iterations from 100 to 500; such specific configurations in the considered experiment are those of #1–#4, #6, #11–#13. And for some other configurations, such as configurations #7, #8 and #14, even though some improvement is made from iteration 100 to 500, the majority of the improvement has already been attained during the first 100 iterations. A clearer understanding and appreciation of the above remarks can be obtained through Figure 5.1, where the performance differences between the current solutions (η_{100} or η_{500}) and the initial solution (η_0) are shown as the columns named "improvements". Thus, a naturally arising and interesting problem is how to identify conditions like those described above, and stop the algorithm before the completion of the pre-specified iterations. It is also important to notice that the availability of such a test will also help guarding against the opposite effect where the algorithm is terminated prematurely, i.e., before getting close to a local optimum, due to an insufficient iteration budget. Finally, we must also notice that the sought test should not rely on an estimation of the objective values reported in Table 5.1,

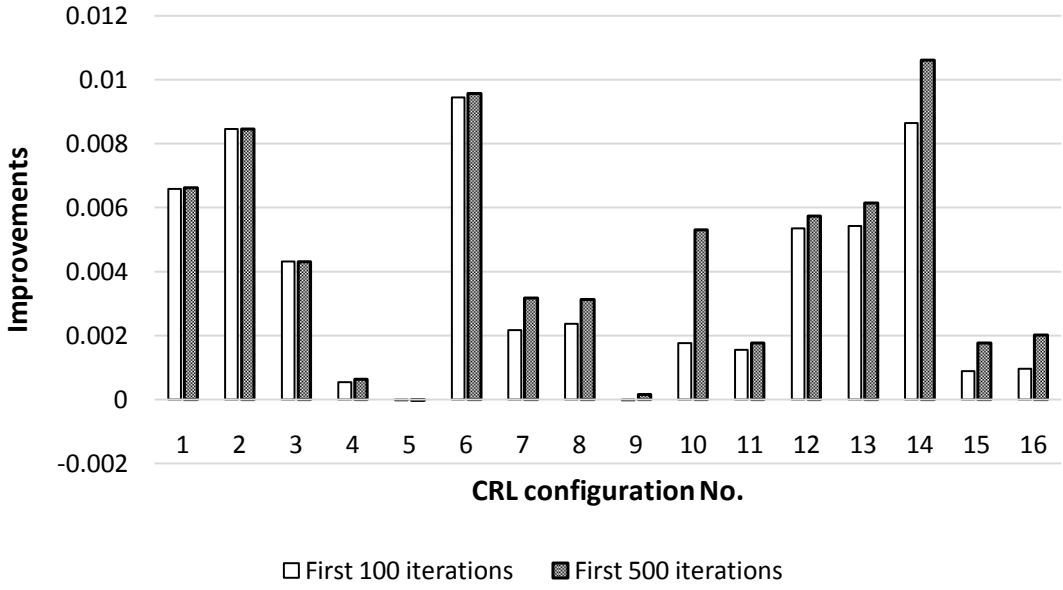


Figure 5.1: Comparison of the throughput improvements made through the first 100 and 500 iterations of Algorithm 5 for the 16 CRL configurations

since, for large(r) RAS configurations, the employed simulations might not be able to return sufficiently precise estimates for these values.

The second issue that is revealed by the data of Table 5.1 regarding the behavior of the baseline SA algorithm, is a particular consumption pattern of computational resources that is reflected in the last two columns of this table. As it can be seen in these columns, the CRL configurations with larger state spaces (c.f. Table 4.2 for the information on the size of these state spaces) usually will take significantly more time for the execution of the same number (500) of iterations. The reason mainly lies on the longer recurrent time for the employed reference marking. As mentioned in the previous paragraph, the timely identification of the access of a "plateau" can save some simulation effort. Additional gains can be achieved by trying to allocate the sample budget more efficiently across the performed iterations. In the aforementioned experiment, the number of regenerative cycles at each iteration is uniformly 10,000. But it is possible that at some iterations fewer regenerative cycles are adequate for obtaining a good estimation of the improvement direction, while at

some other iterations more iterations are actually needed.

In the next two subsections, we address each of the two issues described above, starting with the second one.

5.3.2 Sample-size control in the estimation of the improvement direction

Adaptive sample size selection schemes for sampling-based and simulation optimization algorithms is an issue that has received certain attention within the simulation optimization and the machine learning communities. Hence, for instance, the works of [45, 7, 26] have proposed adaptive sample size selection schemes that are appropriate for the computational context of sample average approximation (SAA) algorithms – these are algorithms that try to solve a stochastic optimization problem by constructing, through adequate sampling, a surrogate deterministic version of the problem and eventually solving this new problem through standard mathematical programming methods. On the other hand, the particular sampling problem that is of interest in this work – i.e., the adaptive determination of the sample size to be employed by the gradient estimator so that the obtained estimate can really improve the current solution with a certain probability – can be addressed through the analysis and the results that are presented in [95, 15]. More recently, the work of [80, 41] has sought the development of a more comprehensive scheme for the dynamic management of the various parameters that define the considered SA recursion, including the employed step size schedule, the sample size scheme, and the stopping criterion; the initial results seem interesting and promising, but more work needs to be done for a complete specification of this method.

In this section, we integrate adaptive sampling capability to the basic SA recursion (31) that was presented in the previous sections of this chapter, by employing and adapting results similar to those developed in [95] and [15]. More specifically, for the sample average Y_{A}^{K} of the improvement direction that is obtained from Equation (46),

1. Given a tentative sample size $N = N_0$, where $N_0/2K$ is an integer, run the simulation for N_0 regenerative cycles, and compute some sample statistics, such as the sample average Y_{A} and the sample covariance $\hat{\Sigma}$.

2. Compute a lower bound N_{new} of the sample size according to the sample statistics. If $N_{\mathrm{new}} \leq N/2K$, then output Y_{A} as the final result for the improvement direction; otherwise continue to the next step.

3. Run the simulation for additional $(2KN_{\mathrm{new}} - N)$ regenerative cycles, update the sample statistics accordingly, replace N with $2KN_{\mathrm{new}}$, and then repeat Step 2 that computes an updated version of N_{new} and performs the tests with the updated data.

Figure 5.2: A generic scheme for sample-size control in the computation of the sample average Y_{A}

there is a relationship between the sample size $N/2K$ and the variation of Y_{A}^K; and with a fixed K, N can be seen as a control parameter that determines the variation – or the "noise" – of Y_{A}^K. Hence, the topic of this section is the determination of N in order to control the noise level in the estimates Y_{A}^K, so that they can lead robustly to the correct improvement direction. In the sequel, the superscript K for the sample average Y_{A}^K and the single sample $\tilde{Y}_i^K$ will be suppressed whenever there is no risk of confusion.

In general, the procedures for sample-size control in the computation of the sample average Y_{A} can be summarized in the steps presented in Figure 5.2. These three steps will be integrated in the adapted version of Algorithm 5, and the resulting algorithm is presented as Algorithm 6 in Section 5.3.4. In the rest of this subsection we discuss the necessary implementational details for the three steps that are depicted in Figure 5.2.

We start by observing that for practical purposes, one must impose a limit for the loop between Steps #2 and #3 in the generic sample-size control scheme that is presented in Figure 5.2. In the presented work, this limit is realized by setting an upper bound $t_{\max}$ on the total number of transitions that are allowed for each

iteration.

Also, in the sequel, the computation of the lower bound N_{new} of the necessary sample size that takes place in Step #2 of the procedure in Figure 5.2, will be based on the available values for the sample average Y_{A} and the sample covariance $\hat{\Sigma}$; and these last two quantities will be maintained in a recursive manner.

More specifically, for the recursion of the sample average, let $Y_{\text{A}}(N_{\text{g}})$ denote the sample average from $2KN_{\text{g}}$ regenerative cycles, or N_{g} samples of $\tilde{Y}_i$. Then, the iterative update is as follows:

$$Y_{\text{A}}(N_{\text{g}}+1) = \frac{N_{\text{g}}}{N_{\text{g}}+1} Y_{\text{A}}(N_{\text{g}}) + \frac{1}{N_{\text{g}}+1} \tilde{Y}_{N_{\text{g}}+1} \tag{55}$$

In the case of the sample covariance, the corresponding update cannot be expressed as a simple iteration. Instead, we can keep track of another iteratively updated sample statistic - the matrix $Y_{\text{Sq}}(N_{\text{g}})$ defined by

$$Y_{\text{Sq}}(N_{\text{g}}) = \sum_{i=1}^{N_{\text{g}}} (\tilde{Y}_i) \cdot (\tilde{Y}_i)^T \tag{56}$$

This last statistic is updated by

$$Y_{\text{Sq}}(N_{\text{g}}+1) = Y_{\text{Sq}}(N_{\text{g}}) + (\tilde{Y}_{N_{\text{g}}+1}) \cdot (\tilde{Y}_{N_{\text{g}}+1})^T \tag{57}$$

and the sample covariance can be updated as follows:

$$\hat{\Sigma} = \frac{Y_{\text{Sq}}(N_{\text{g}}) - N_{\text{g}}(Y_{\text{A}}(N_{\text{g}})) \cdot (Y_{\text{A}}(N_{\text{g}}))^T}{N_{\text{g}}-1} \tag{58}$$

With the estimates Y_{A} and $\hat{\Sigma}$ readily available, next we consider different methods to define and calculate the lower bound of the sample size, N_{new}, in Step 2 of Figure 5.2.

In [95], N_{new} is established based on the notion of the "confidence region". According to the Central Limit Theorem [71], the sample average Y_{A} is approximately multi-normally distributed if the sample size is big enough. Then, this fact can be

used for constructing confidence regions that will contain the vector $\mathbb{E}[Y_{\mathrm{A}}]$, that defines the gradient direction, with some specified probability. Furthermore, for the direction of Y_{A} to be an ascending direction, the angle between the vectors Y_{A} and $\mathbb{E}[Y_{\mathrm{A}}]$ must be acute, i.e., the inner product $Y_{\mathrm{A}}^T \cdot \mathbb{E}[Y_{\mathrm{A}}]$ must be positive. Hence, the sampling method of [95] first determines a certain region for $\mathbb{E}[Y_{\mathrm{A}}]$ that would satisfy this acuteness condition for the angle of the vectors Y_{A} and $\mathbb{E}[Y_{\mathrm{A}}]$, and subsequently computes the required sample size N_{new} so that the aforementioned region will be a confidence region for $\mathbb{E}[Y_{\mathrm{A}}]$ with a specified level of confidence.

On the other hand, the work of [15] seeks to satisfy a similar "acuteness" condition for the angle of the vectors Y_{A} and $\mathbb{E}[Y_{\mathrm{A}}]$, but it tries to determine the minimal required sample size N_{new} through a different line of analysis. More specifically, the approach proposed in [15] is motivated from the following sufficient condition for the vector Y_{A} to be an improvement direction: The Euclidean distance between the estimated and the true gradients should be no larger than a scaled Euclidean norm of the estimated gradient with the scaling factor less than one. In the context of our sampling problem, this condition translates to:

$$||Y_{\mathrm{A}} - \mathbb{E}[Y_{\mathrm{A}}]|| \leq \theta ||Y_{\mathrm{A}}||, \quad 0 \leq \theta < 1$$

Equivalently,

$$||Y_{\mathrm{A}} - \mathbb{E}[Y_{\mathrm{A}}]||^2 \leq \theta^2 ||Y_{\mathrm{A}}||^2, \quad 0 \leq \theta < 1 \tag{59}$$

But since the vector $\mathbb{E}[Y_{\mathrm{A}}]$ is an unknown quantity, the authors of [15] propose to use the l_1-norm of the "component-wise" variance estimator in the place of the left-hand-side of Equation (59), and the corresponding condition is redefined as follows:

$$
\begin{aligned}
||Y_{\mathrm{A}} - \mathbb{E}[Y_{\mathrm{A}}]||^2 &\approx \mathbb{E}\left[||Y_{\mathrm{A}} - \mathbb{E}[Y_{\mathrm{A}}]||^2\right] \\
&= ||\mathrm{VAR}(Y_{\mathrm{A}})||_1 \\
&= \sum_{j=1}^{p} \mathrm{VAR}(Y_{\mathrm{A}}[j]) \\
&= \frac{1}{N_{\mathrm{new}}} \sum_{j=1}^{p} \mathrm{VAR}(\tilde{Y}_i[j]) \\
&\approx \frac{1}{N_{\mathrm{new}}} \sum_{j=1}^{p} \frac{1}{N_0/2K - 1} \sum_{i=1}^{N_0/2K} (\tilde{Y}_i[j] - Y_{\mathrm{A}}[j])^2 \\
&\leq \theta^2 ||Y_{\mathrm{A}}||^2, \quad 0 \leq \theta < 1
\end{aligned} \tag{60}
$$

A drawback of the work in [15] is that it does not quantify the chance for potential error that is brought about by the above approximations; i.e., there is a non-negligible but unknown probability that the last inequality of (60) holds but the original inequality of (59) does not hold. From this standpoint, the method of [95] is a more attractive choice for our application context, since this method establishes an explicit level of confidence that Y_{A} is an ascending direction. As a result, in the following we shall focus on the method of [95], which will be referred to as the *Shapiro-Mello* sample-size control method (based on the last names of its originators).

On the other hand, while defining clearly the confidence for a "correct" improvement direction, the Shapiro-Mello method can result in very high sample sizes in regions where the norm of the estimate Y_{A} is quite small. This fact will be revealed in the following technical deliberations, and it is also manifested in the results of a numerical study that is presented in a later part of this section. The latter also shows a large variation in the resultant N_{new} values that are computed with this method. These effects are caused by the acuteness requirement on the angle between the estimator Y_{A} and $\mathbb{E}[Y_{\mathrm{A}}]$, which is used to control the variance of Y_{A} as well as the direction and the norm of the estimate itself. In response to these challenges, two other methods are also introduced in the sequel. These two methods solely control the level of

the variance of $Y_{\rm A}$; for this reason, they will be collectively categorized as *variance-based* sample-size controls in the sequel.[9] As it will be explained in the following, the variance-based methods essentially will tolerate an arbitrarily large probability of selecting a "wrong" direction in certain parts of the underlying solution space (i.e., a direction $Y_{\rm A}$ that fails to form an acute angle with the vector $\mathbb{E}[Y_{\rm A}]$), in an effort to control the variability of the sampling effort that is required for the computation of the estimator $Y_{\rm A}$ at the different points $\bar{\zeta}$ that are visited by the SA algorithm. This negative effect is moderated by a single parameter that is employed in the design of the variance-based methods, and acts as a scaling mechanism that specifies the discriminatory power of the method. From a more conceptual standpoint, the relaxed resolution / discriminatory power of the proposed variance-based methods is motivated by the realization that the actual loss of this power will take place at points where the norm of the vector $Y_{\rm A}$ is pretty small, and therefore, a potential move in an erroneous direction will not have a very significant impact on the overall progress of the underlying SA algorithm. Next, we proceed with a detailed description of the three methods and the underlying motivation for each of them.

The Shapiro-Mello method The Shapiro-Mello sample-size control method was first proposed in [95] and it is discussed in higher detail in [44]. Here, we will outline the main developmental guidelines of this method, and customize it to the estimator of the improvement direction that was developed in Section 5.2.2; for a complete exposure of the method, the reader is referred to Section 4, Chapter 3 of [44].

As already pointed out in the previous discussion, for a gradient estimator $\widehat{\nabla\eta}$ and a given confidence level $1-\alpha$, we can build an ellipsoid as the *confidence region* for the true value $\nabla\eta$. Furthermore, there is a relationship between the size of the confidence region and the sample size that was employed in the computation of the

[9] On the other hand, the original Shapiro-Mello method and the method of [15] can be categorized as *angle-based* sample-size controls.

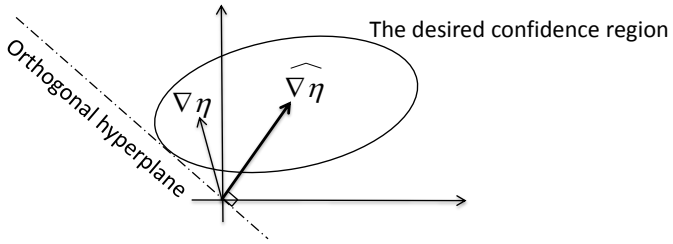


Figure 5.3: The gradient estimate $\widehat{\nabla\eta}$, the true gradient vector $\nabla\eta$, and the desired confidence region (adapted from [44])

estimate $\widehat{\nabla\eta}$. The Shapiro-Mello sample-size control method determines the lower bound N_{new} so that the ellipsoid of the corresponding confidence region is tangent with the hyperplane that is orthogonal to the gradient estimate $\widehat{\nabla\eta}$. The relationship between the gradient estimate $\widehat{\nabla\eta}$, the true value $\nabla\eta$, and the ellipsoid of the confidence region obtained with the lower-bound sampling size N_{new} is depicted in Figure 5.3.

In more specific terms, suppose that there are n independent and identically distributed random vectors v_i with dimension p, and their mean and covariance are respectively μ and Σ, where Σ is positive definite. Then, according to the theoretical developments in [69], as $n \to \infty$:

$$n \cdot (\bar{v}_n - \mu)^T \cdot \Sigma^{-1} \cdot (\bar{v}_n - \mu) \xrightarrow{D} \chi_p^2 \tag{61}$$

In Equation (61), $\bar{v}_n$ is the sample average of the vectors $v_i,\ i = 1, \ldots, n$, the notation " $\xrightarrow{D}$ " denotes converge in distribution, and χ_p^2 is the chi-squared distribution with p degrees of freedom.

From Equation (61), a $100(1-\alpha)\%$ confidence region for μ can be represented by the ellipsoid

$$E_\alpha(n) \equiv \left\{ x \in \mathbb{R}^p : (x - \bar{v}_n)^T \cdot \Sigma^{-1} \cdot (x - \bar{v}_n) \leq \frac{\chi_p^2(\alpha)}{n} \right\} \tag{62}$$

where $\chi_p^2(\alpha)$ is the right-tail critical value for the chi-squared distribution, i.e., $\mathbb{P}\{\chi_p^2 > \chi_p^2(\alpha)\} \equiv \alpha$.

Furthermore, the aforementioned requirement for an acute angle between the vectors $\bar{v}_n$ and μ can be expressed as follows:

$$\bar{v}_n^T \cdot x \geq 0, \ \forall x \in E_\alpha(n) \tag{63}$$

Finally, the condition of Equation (63) can be converted to the following specification for the sample size n [44]:

$$n \geq \left\lceil \chi_p^2(\alpha) \cdot \frac{\bar{v}_n^T \cdot \Sigma \cdot \bar{v}_n}{(\bar{v}_n^T \cdot \bar{v}_n)^2} \right\rceil \tag{64}$$

Although in the original method of [44] the lower bound for the sample size n that is specified in Equation (64) is only applied to the gradient estimator, the method is also applicable to other estimators with their expectations proportional to the gradient. In the case of the estimator Y_{A} of Equation (47), the implementation of the above method requires the covariance matrix Σ of the single sample $\tilde{Y}_i$. But since this matrix is not available, we shall replace it by the estimated covariance matrix $\hat{\Sigma}$. Then, Equation (64) can be customized to the estimation of the lower bound N_{new}, as follows:

$$N_{\mathrm{new}} = \left\lceil \chi_p^2(\alpha) \cdot \frac{Y_{\mathrm{A}}^T \cdot \hat{\Sigma} \cdot Y_{\mathrm{A}}}{\left(Y_{\mathrm{A}}^T \cdot Y_{\mathrm{A}}\right)^2} \right\rceil \tag{65}$$

The lower bound N_{new} that is specified by Equation (65), is a function of the sample statistics that are obtained from the regenerative cycles of the running simulation of the underlying GSPN. As a result, N_{new} itself is a random variable with a certain variance. Next, we assume a known covariance matrix Σ and we obtain a lower bound for the expected value of N_{new}. For that, first we notice that for the given covariance matrix Σ, N_{new} can be written as

$$\begin{aligned} N_{\mathrm{new}} &= \left\lceil \chi_p^2(\alpha) \cdot \frac{Y_{\mathrm{A}}^T \cdot \Sigma \cdot Y_{\mathrm{A}}}{\left(Y_{\mathrm{A}}^T \cdot Y_{\mathrm{A}}\right)^2} \right\rceil \\ &= \left\lceil \frac{\chi_p^2(\alpha)}{||Y_{\mathrm{A}}||^2} \cdot \left(\frac{Y_{\mathrm{A}}^T}{||Y_{\mathrm{A}}||} \cdot \Sigma \cdot \frac{Y_{\mathrm{A}}}{||Y_{\mathrm{A}}||} \right) \right\rceil \end{aligned} \tag{66}$$

where $||Y_{\rm A}||$ denotes the Euclidean norm of the vector $Y_{\rm A}$. Let $\hat{\rho} > 0$ denote the minimal eigenvalue of matrix Σ.[10] Then, since $Y_{\rm A}/||Y_{\rm A}||$ is a unit-norm vector, we have that

$$\frac{Y_{\rm A}^T}{||Y_{\rm A}||} \cdot \Sigma \cdot \frac{Y_{\rm A}}{||Y_{\rm A}||} \;\geq\; \hat{\rho} \tag{67}$$

and therefore,

$$\begin{aligned} \mathbb{E}[N_{\rm new}] &\geq \chi^2_p(\alpha) \cdot \hat{\rho} \cdot \mathbb{E}\left[\frac{1}{||Y_{\rm A}||^2}\right] \\ &\geq \chi^2_p(\alpha) \cdot \hat{\rho} \cdot \frac{1}{\mathbb{E}\left[||Y_{\rm A}||\right]^2} \end{aligned} \tag{68}$$

The last inequality above is an application of Jensen's inequality [92]. Equation (68) implies that the amount of sampling specified by $N_{\rm new}$ will be particularly high for policies $\bar{\zeta}$ that are in the region of an (interior) local optimum, and therefore, the magnitude of the gradient vector $\nabla\eta$ is quite small. In fact, at such points of the algorithm evolution, the sampling requirements may not be practically feasible within the scope of the available computational resources.

Besides the possibility of a very large magnitude of $N_{\rm new}$, another issue of concern with the Shapiro-Mello method that was outlined in the previous paragraphs, is the potential large variations of $N_{\rm new}$. This variation results from the presence of the inner product of the multi-normally distributed quantity $Y_{\rm A}$ at the denominator of Equation (66), and it can even lead to an infinite expectation for the random variable $N_{\rm new}$ when the dimension of the underlying solution space is $p = 1$. More specifically, when the problem consists of only one decision variable, the right-hand-side of Equation (65) approximately degenerates to the square of a reciprocal normally distributed random variable (cf., page 171 of [49]), or equivalently, an inverted-chi-squared distributed random variable (cf., pages 119 and 431 of [10]) with one degree of freedom; and such a random variable is known to have no expectation.[11] The numerical study that is

[10] We remind the reader that the covariance matrix Σ has been assumed positive definite.

[11] A reciprocal normally distributed random variable is a random variable obtained by taking the

presented in a later part of this section will also demonstrate the possibility for such a divergent behavior.

The aforementioned remarks, together with the realization that, in areas where the response surface is rather flat, small steps (even in the wrong direction) do not have a particularly strong impact on the performance of the underlying algorithm, motivates the variance-based methods for the determination of the sample size N_{new} that are discussed next.

Variance-based sample-size control As discussed in the previous paragraphs, variance-based sample-size methods are employed in an effort to control the potential explosion and the high variation of the sample sizes that are specified by the Shapiro-Mello method. In order to effect this control, let us consider the formula that provides the volume V of the confidence ellipsoid $E_\alpha(n)$ defined by Equation (62). This formula is (c.f. Section 3.1 in [40]):

$$V = C(p)(\det \Sigma)^{1/2} \left(\frac{\chi_p^2(\alpha)}{n} \right)^{p/2} \tag{69}$$

where $\det \Sigma$ is the determinant of matrix Σ, and $C(p)$ is the volume of a p-dimensional ball with unit radius, i.e.,

$$C(p) \equiv \frac{\pi^{p/2}}{\Gamma\left(\frac{p}{2}+1\right)} = \begin{cases} \frac{\pi^k}{k!} & p = 2k, k = 1, 2, \ldots \\ \frac{2(k!)(4\pi)^k}{(2k+1)!} & p = 2k+1, k = 0, 1, 2, \ldots \end{cases} \tag{70}$$

Suppose that we impose a limit for the maximum possible volume for these confidence ellipsoids equal to $V_{\max}$. Then, from Equation (69) we get:

$$n \geq \left\lceil \chi_p^2(\alpha) \left(\frac{C(p)^2 \ \det \Sigma}{V_{\max}^2} \right)^{1/p} \right\rceil \tag{71}$$

reciprocal of a normally distributed random variable. An inverted-chi-squared distributed random variable is a random variable obtained by taking the reciprocal of a chi-squared distributed random variable. The expectation for N_{new} in case of $p = 1$ does not exist, since for a normally distributed random variable with mean μ and variance σ^2, the integral $\int_{-\infty}^{\infty} \exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right) \frac{1}{x^2} dx$ does not converge.

The right-hand-side in Equation (71) is the smallest sample size that, with probability $1-\alpha$, will keep the point that is defined by the gradient vector $\nabla\eta$ within an ellipsoid of volume $V_{\max}$ and shape defined by the covariance matrix Σ, that is centered at the point defined by the corresponding gradient estimate $\widehat{\nabla\eta}$.

In the context of the gradient estimation problem that is considered in this section, an important property of the sample size that is defined by the right-hand-side of Equation (71) is that it is independent from the estimate Y_{A}. In particular, Equation (71) implies that, for any given policy $\bar{\zeta}$, the aforementioned sample size will depend only upon the determinant of the corresponding covariance matrix Σ, or equivalently, upon the product of all the eigenvalues of Σ [70]; these eigenvalues correspond to the variances of the projections of the vectors Y_{A} on the p axes of the considered ellipsoid.

Of course, as in the case of the Shapiro-Mello sample-size control method, in order to apply the result of Equation (71) to the estimation problem considered in this section (i.e., the construction of the estimator Y_{A} in Equation (47)), we need to substitute the covariance matrix Σ with the sample covariance $\hat{\Sigma}$, a fact that will increase the variability in the obtained estimates for N_{new}. We should also notice that the corresponding ellipsoid essentially constitutes a confidence region for the vector $2K^2\mathbb{E}[\tau_1]^2\nabla\eta$, which defines $E[Y_{\mathrm{A}}]$ and depends on $\mathbb{E}[\tau_1]$ as well as on $\nabla\eta$. And the space dimensions, p, are different for different RAS since the numbers of the decision variables involved will be different. Therefore, instead of imposing an explicit limit on the volume of the ellipsoid that represents the confidence region as discussed above, we propose to define this quantity in more implicit terms, by specifying a control parameter that pertains to a single dimension. Let us denote this parameter as l_0. Then, considering (i) the fact that our algorithm will eventually use the estimate $\widehat{\nabla\eta}$ for the specification of the improvement direction, and (ii) the scaling factor $2K^2\mathbb{E}[\tau_1]^2$ that relates the mean of the estimator Y_{A} and $\widehat{\nabla\eta}$, we also define the volume $(2l_0K^2\mathbb{E}[\tau_1]^2)^p$ as a "nominal" volume that is induced by the selection of

the parameter l_0, and eventually we use this last quantity as the maximal volume $V_{\max}$. Furthermore, during the actual evaluation of this volume, the expectation $\mathbb{E}[\tau_1]$ must be replaced by its estimator τ_N/N, where N is the total number of regenerative cycles that is used to obtain the sample statistics Y_{A} and $\hat{\Sigma}$. Then, when adapted to the above developments, Equation (71) gives the following estimation of the sought lower bound N_{new}:

$$N_{\mathrm{new}} = \left\lceil \frac{\chi_p^2(\alpha)}{\left(2l_0K^2\tau_N^2/N^2\right)^2} \left(C(p)^2 \ \det\hat{\Sigma}\right)^{1/p} \right\rceil \tag{72}$$

In general, the calculation of the determinant of a dense p-dimensional matrix has a complexity of $O(p!)$. Obviously, this complexity becomes unacceptable even if p is at a relatively small level (like several tens). This remark motivates the third method for the computation of the sought lower bound N_{new}, that can be perceived as a simplifying scheme for the previous method. More specifically, this approach will ignore the non-diagonal elements of the covariance matrix Σ and approximate the ellipsoids considered by the previous method as cuboids. Let $\bar{\mathrm{v}}$ be the vector consisting of the diagonal components of Σ. Then, this new method will substitute the original covariance matrix Σ by the diagonal matrix $V \equiv \mathrm{diag}[\bar{\mathrm{v}}[i],\ i = 1, \ldots, p]$.

The ellipsoid that is defined by the matrix V has its axes parallel to the axes of the original coordinate system, and each diagonal element $\bar{\mathrm{v}}[i],\ i = 1, \ldots, p$, defines the variance of the estimated vector with respect to the corresponding coordinate axis. Hence, following a logic similar to that used in the specification of the previous variance-based method, for a given confidence level $1 - \alpha$, we may seek to confine the maximum length of the confidence intervals that are built for each component of the estimated vector to a certain level $l_{\max} \equiv 2K^2\mathbb{E}[\tau_1]^2 l_0$; i.e., we can try to select the sample size n in a way that satisfies

$$z(\alpha'/2)\sqrt{\frac{1}{n}||\bar{\mathrm{v}}||_\infty} \ \leq \ \frac{l_{\max}}{2} \tag{73}$$

where $\alpha' \equiv 1 - (1 - \alpha)^{1/p}$, and $z(\alpha')$ is the right-tail critical value for the standard

normally distributed random variable Z (i.e., $\mathbb{P}\{Z > z(\alpha')\} \equiv \alpha'$).

Then, using the diagonal elements of the sample covariance matrix $\hat{\Sigma}$ as an estimate of $\bar{\mathrm{v}}$, to be denoted by $\hat{\mathrm{v}}$, and also taking into consideration the scaling aspects that were discussed in the presentation of the previous method regarding the determination of the limiting volume $V_{\max}$, we can obtain the following value for the lower bound N_{new} from Equation (73):

$$\begin{aligned} N_{\text{new}} &= \left\lceil \left(\frac{z(\alpha'/2)}{l_0 K^2 \tau_N^2 / N^2} \right)^2 ||\hat{\mathrm{v}}||_\infty \right\rceil \\ &= \left\lceil \left(\frac{z(\alpha'/2)}{l_0 K^2 \tau_N^2 / N^2} \right)^2 \max_{i=1,\ldots,p} \left| \hat{\mathrm{v}}[i] \right| \right\rceil \end{aligned} \tag{74}$$

Remark Both Equations (72) and (74) define the corresponding estimates of N_{new} as functions of the sample covariance $\hat{\Sigma}$ and of the stopping time τ_N, and this fact determines the stochastic nature of these quantities. Yet, the lower bound obtained from Equation (74) is expected to be more demanding, in terms of the stipulated amount of sampling, than the lower bound obtained from Equation (72), since Equation (74) essentially confines the ellipsoid that represents the confidence region for the gradient estimator into a cube. More specifically, if all the parameters in Equations (72) and (74) are set at the same values, then the value for N_{new} specified by Equation (74) should be statistically larger than the corresponding value that is specified by Equation (72). This effect can be validated by observing the $\hat{\mu}(N_{\text{new}})$ data of Tables B.3 and B.4 in Appendix B: The reader can see that the $\hat{\mu}(N_{\text{new}})$ values reported in Table B.4 are always greater than the corresponding $\hat{\mu}(N_{\text{new}})$ values reported in Table B.3.

A numerical comparison of the three sample-size control methods As already mentioned, all the three methods for sample-size control that are defined by Equations (65), (72) and (74) specify the lower bound N_{new} as a random variable

with a certain variance. To investigate the variability in the estimates that are generated by each method, we performed a numerical experiment on the first 16 smaller CRL configurations of Table 4.1, while maintaining pretty much the same settings that were employed in the "baseline" experiment of Section 5.3.1. But this time, we preselect an applied policy $\bar{\zeta}$ and a sample-size control method, and while keeping the policy fixed, we run $100N_0$ regenerative cycles of the underlying simulation in order to draw 100 samples of the corresponding random variable N_{new}. In these runs, the significance level for the confidence regions and the confidence intervals involved was set to $\alpha = 0.2$, and the tentative number of regenerative cycles to $N_0 = 10,000$. Also, as in the baseline experiment, the group size K applied in the generation of the samples $\tilde{Y}_i$ was set to 100 pairs of regenerative cycles. Hence, the tentative initial sample size is $N_0/2K = 50$, which is sufficiently large for the invocation of the Central Limit Theorem. Finally, for the variance-based methods, the control parameter l_0 was set to $l_0 = 0.01$.

Each data entry, or *observation*, of the experimental output is classified by three factors: the CRL configuration, the applied policy $\bar{\zeta}$, and the sample-size control method. In particular, for each of the 16 CRL configurations, we perform our experiment on two policies and the three sample-size selection methods that were introduced in this section. The two employed policies include the totally random policy $\bar{\zeta}_0$ that was defined in the baseline experiment in Section 5.3.1, and the "near-optimal" solution characterized as policy $\bar{\zeta}_{500}$ in the previous experiment; the two policies will be labeled as "Point 1" and "Point 2" in the following. The three methods under test are: the Shapiro-Mello control method, labeled as "S-M"; the variance-based method that restricts the volume of the confidence ellipsoid, labeled as "Volume"; and the method that restricts the maximum length of the confidence intervals for each component of the gradient estimate, labeled as "Max CI".

Since this experiment is designed to draw conclusions on the variability of the

Table 5.2: Comparison of the coefficients of variation for the three sample-size control methods on the first 16 CRL configurations of Table 4.1

Conf.	Point 1 (complete random)			Point 2 (near-optimal)		
	S-M	Volume	Max CI	S-M	Volume	Max CI
1	0.77705	0.15117	0.19005	3.13572	0.18885	0.24182
2	0.59753	0.18707	0.20066	8.40951	0.22600	0.22256
3	3.06774	0.17485	0.25644	4.28964	0.17540	0.24144
4	2.00966	0.14525	0.20865	3.74781	0.16341	0.23769
5	5.53503	0.18464	0.24502	1.90922	0.15793	0.24996
6	1.83828	0.20258	0.18668	9.95461	0.24879	0.24605
7	1.59014	0.23626	0.25452	4.90151	0.17988	0.25282
8	2.68928	0.20825	0.30671	5.78407	0.20716	0.30961
9	2.98093	0.27938	0.37871	3.04971	0.25818	0.34605
10	0.47452	0.11054	0.28324	0.66420	0.09725	0.24652
11	1.07488	0.11040	0.16024	3.55511	0.10817	0.20616
12	1.08522	0.12463	0.19949	1.01185	0.10465	0.17375
13	0.75306	0.09921	0.15596	0.94941	0.10294	0.18427
14	0.71545	0.13483	0.16997	1.21458	0.12810	0.21715
15	1.15613	0.11397	0.19971	0.65112	0.11213	0.23033
16	0.74673	0.10098	0.14700	0.67287	0.09019	0.18162
Avg.	1.69323	0.16025	0.22144	3.36881	0.15931	0.23674

N_{new} estimates that are generated by the considered methods at different operational settings of the SA algorithm, first we consider the sample coefficient of variation (i.e., the sample standard deviation divided by the sample average) for N_{new} at each setting. The corresponding results are presented in Table 5.2, while Tables B.2–B.4 in Appendix B also report the values of the corresponding sample means and standard deviations.

Table 5.2 reports the coefficients of variation, itemized by the CRL configurations, the employed sample-size selection methods, and the applied policies. In general, the variability observed in the Shapiro-Mello method is larger than the variability that is observed in the variance-based sample-size control methods. On the other hand, when comparing the reported results for the two variance-based methods, we can see that the omission of the non-diagonal elements of the covariance matrices Σ in the "Max CI" method, generally causes an increase of the variability, but the corresponding

Table 5.3: Comparison of the growth of the N_{new} estimates between the two policies $\bar{\zeta}_0$ and $\bar{\zeta}_{500}$ for the three sample-size control methods on the first 16 CRL configurations of Table 4.1

Conf.	S-M	Volume	Max CI
1	31.39467	4.26153	3.33152
2	49.59404	10.89854	9.68864
3	1.30993	2.80268	2.06336
4	5.17075	1.21479	1.48393
5	0.18823	0.44476	0.50893
6	86936.37387	6.17236	5.81308
7	0.33870	2.69329	3.15603
8	2.32290	2.86709	2.70935
9	1.05174	0.81078	1.01438
10	1.79780	1.84603	5.38172
11	3.64390	1.25065	1.39112
12	2.74744	2.17358	3.17267
13	7.23524	2.50105	4.13061
14	5.11668	5.78965	8.56895
15	0.99184	1.31295	2.03813
16	1.76763	1.66969	3.39030
Avg.	5440.69034	3.04434	3.61517

coefficients of variation are still quite small (below 0.4). Hence when considered from this standpoint, the variance-based selection methods have an advantage over the Shapiro-Mello control method, while the omission of the non-diagonal elements of Σ by the "Max CI" method does not define any substantial disadvantage for it with respect to the "volume"-based method.

Table 5.3 reports the ratios of the sample sizes N_{new} generated at each of the two points $\bar{\zeta}_0$ and $\bar{\zeta}_{500}$, for each sample-size selection method and each CRL configuration. From the results that are reported in this table, we can see that, in most CRL configurations, the required sample sizes increase as the SA algorithm proceeds closer to the near-optimal regions; in particular, all three methods generally require more samples at Point 2. Furthermore, the sample sizes specified by the Shapiro-Mello method at this point are more variable than those specified by the variance-based

methods.[12]

An intuitive explanation for the increase of the sampling requirements in the case of solutions that are in near-optimal regions, has as follows: When the current solution of the SA algorithm is approaching to a plateau, the selection probability distributions of the static random switches are more "biased" to certain options. Therefore, the steady-state distribution of the underlying DTMC is also highly biased towards certain states with higher one-step immediate rewards, and although the DTMC is still irreducible (due to the imposed randomization), some states are rarely visited. Then, the behavior of the DTMC in each possible sample path that can be followed in a single regenerative cycle can be very different, and this variability is also reflected in the higher variance of the estimates $\tilde{Y}_i$ that are obtained from the simulation of these sample paths. Since all the three methods are dependent on the covariance matrix Σ for the determination of the specified N_{new}, they will tend to return larger values for N_{new} at the aforementioned points.

Of course, for the Shapiro-Mello method, Equation (68) provides an additional explanation for the high sampling requirements that are posed by this method for points in near-optimal regions. Figure 5.4 illustrates this last reason for the larger ratio of the Shapiro-Mello method in Table 5.3 in a more intuitive way, with an example case of two decision variables x_1 and x_2. In both the left and right figures, the gradient estimates are depicted in boldfaced arrows, and the one on the left figure

[12]The reader may notice that CRL configuration #6 has an extremely large ratio under the Shapiro-Mello method. This is due to some very large N_{new} values obtained at Point 2. The reason is the unbounded expectation of N_{new}, since the corresponding scheduling problem involves only one decision variable (c.f. the corresponding remarks at the end of the description of the Shapiro-Mello method). As shown in Table 4.3 in Section 4.2.2, the CRL configurations #2 and #6 have only one decision variable, so they are susceptible to experiencing some extremely large N_{new} values when the Shapiro-Mello method is used in the corresponding SA algorithm. And the quantity of $||\nabla\eta||$ at Point 2 of CRL configuration #6 is the smallest among the corresponding values at Points 1 and 2 for CRL configurations #2 and #6; this, in turn, implies a lower value of $\mathbb{E}[Y_{\text{A}}]$ and a higher risk of some extremely large values for N_{new}. On the other hand, for problem instances with a higher dimension for their solution space, the distribution of N_{new} is similar to an inverted-chi-squared distribution with higher degrees of freedom, whose expectation exists.

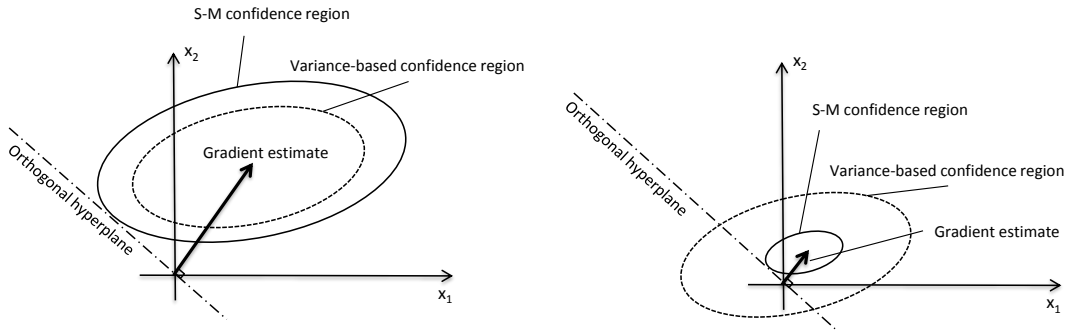


Figure 5.4: Comparison of the confidence regions specified by the Shapiro-Mello and the variance-based methods for different norms of the gradient estimates.

has a larger norm. The confidence regions required by the variance-based methods have the same areas, which are depicted in the ellipses with dashed lines. On the other hand, the confidence regions required by the Shapiro-Mello method, which are depicted in the ellipses with solid lines, should always be tangent with the hyperplanes orthogonal to the gradient estimates. Therefore, the area of this ellipse on the left figure is larger, which implies a more relaxed requirement on the corresponding sample size than the respective requirement specified by the constant ellipse; in contrast, the sample size requirement becomes more demanding when the norm of the gradient estimate is smaller.

But as already pointed out, for solutions with small-norm gradient estimate, it is not necessary to keep an acute angle under the conservative constant-step-size schedules considered in this work, since the corresponding step will not move the solution too much. On the other hand, if the gradient estimate has a large norm, then forcing an angle between this estimate and the points of the constructed confidence region that is smaller than "just acute", can give better performance.

Based on all the above remarks and findings, we have chosen the "maximum length of confidence intervals" method as the method to be employed by the proposed Algorithm 6 in our empirical studies. This method presents much less variability in its estimates of N_{new} than the variability that is exhibited by the Shapiro-Mello method,

and furthermore, by avoiding the computation of the determinant of $\hat{\Sigma}$, it is more practical for larger numbers of decision variables. This last feature becomes especially important when Algorithm 6 is applied under the partial disaggregation scheme that is introduced in Section 4.3.

5.3.3 A statistical test for the terminating condition

As we have already seen, from a practical standpoint, an SA algorithm terminates either because the allocated computation resource is exhausted, or the current solution is considered good enough and the computation cost for making further progress is too high. In the latter case, the algorithm is in a plateau of the solution space of the maximization problem. In this section, we are interested in the development of statistical tests able to identify that the current solution is in the plateau.

Some of the earliest efforts for the identification of such conditions have been based on the development of an asymptotic distribution characterizing the algorithm behavior around the sought optima. And they are based on the fact that, as the SA recursion (31) proceeds to a near-optimal region under an appropriately decreasing schedule for the step size $\gamma(n)$, the solutions returned by the algorithm are confined in a small region that can be described as a (time-)non-homogeneous stochastic process $\{\bar{\zeta}_n\}$ with an approximately constant mean and a decreasing variance. Hence, in [96], the decision variables generated in all past iterations, from the very beginning to the current iteration, are used to estimate the asymptotic variance, and the algorithm stops when this variance estimate results in a confidence interval for the current mean that is sufficiently small. Also, the work of [101] developed the multi-dimensional version of the aforementioned method. On the other hand, the work of [46] has improved upon the ideas and methods of [96] and [101] by introducing a "surrogate" process that approximates the distribution of the current solution in a better manner, and therefore, improves the discerning power of the corresponding termination condition.

In the case of the constant step-size algorithms like the one that is eventually proposed in this work, all the aforementioned methods must be adjusted to account for the fact that the variance of the asymptotic distribution characterizing the algorithm motion around the target optimal solution will not decrease to zero but will remain at a constant level. Instead of pursuing this possibility, in this work we seek to recognize the algorithm access of a plateau, or more generally, its proximity to an optimal solution, by considering the distribution of the improving direction Y_n, and developing statistical tests that will ascertain the inability of this vector to define any further substantial progress for the algorithm. In particular, inability to reject the null hypothesis of these tests implies that the variance of the Y_n estimator dominates the true value of the gradients, and the algorithm begins to wander on a plateau.

Termination hypothesis tests based on the KKT optimality condition From a more technical standpoint, the last approach for the development of a terminating condition mentioned above can be based on a pertinent assessment of Condition 7 in Section 5.1. In particular, this assessment must effectively account for the noise that exists in the estimation of the improvement direction. The works of [95, 44] proposed some methods for the development of such a hypothesis test for the gradient estimator employed in the context of the SAA algorithm that is developed in that work. Next, we adapt these methods to the context of the SA algorithm that is pursued in this document.

More specifically, the methods of [95, 44] start with the observation that, for a given solution $\bar{\zeta}^*$, the expression in the right-hand-side of Equation (32) defines a cone, to be called the optimal cone $\mathcal{C}$ in the sequel. Since the true value $\nabla\eta$ is unknown, the null hypothesis is that $\nabla\eta \in \mathcal{C}$; and the test is based on the distance d between the cone $\mathcal{C}$ and the gradient estimate $\widehat{\nabla\eta}$, where $\mathbb{E}[\widehat{\nabla\eta}] \equiv \nabla\eta$. If the null hypothesis is not rejected, then we can infer that the current solution is not distinguishable from

the set of solutions that satisfy the optimality condition. Note that the improvement direction estimator Y_{A} obtained from Equation (44) or (47) can also be put to this test since there exists a positive scalar a such that $E[Y_{\mathrm{A}}] = a\nabla\eta$. Then $\nabla\eta \in \mathcal{C}$ if and only if $E[Y_{\mathrm{A}}] \in \mathcal{C}$. Therefore, in the sequel, the notation Y will replace $\widehat{\nabla\eta}$.

In the following, we present two methods for supporting the test that was described in the previous paragraph, using two different norms for characterizing the involved distance. Both of these methods require an additional assumption for the solution space of the problem that is called the *strict complementarity* assumption, and it is stated as follows:

Assumption 5 *For all local optimal solutions $\bar{\zeta}^* \in H$, the corresponding Lagrange multipliers λ_j in the decomposition of the gradient at this solution, $\nabla\eta(\bar{\zeta}^*)$, along the directions that are defined by the gradients of the binding constraints, are all strictly positive.*

Remark For the problem defined in Equations (33)–(35), the constraints are linearly independent. Thus, Assumption 5 holds if the changes of the decision variables along the gradient directions of any binding constraints changes the objective value. But Assumption 5 will not hold if the objective value is independent from some random switches. The possibility and the effective identification of such cases needs some further investigation, and the decision variables associated to such random switches should be removed during the execution of the considered tests. In the following, we shall assume that the strict complementarity assumption holds at all the points of interest. □

For a given estimated improvement direction Y at a solution $\bar{\zeta}^*$ with estimated covariance $\hat{\Sigma}$, the distance between Y and $\mathcal{C}$ can be expressed as either

$$d_1 \equiv \min_{\lambda \in \mathbb{R}_{0+}^{|B|}} (Y - W\lambda)^T \hat{\Sigma}^{-1} (Y - W\lambda) \tag{75}$$

or

$$d_2 \equiv \min_{\lambda \in \mathbb{R}_{0+}^{|B|}} (Y - W\lambda)^T (Y - W\lambda) \tag{76}$$

where B is the index set of the binding constraints at $\bar{\zeta}^*$, λ is a $|B|$-dimensional vector whose components are the Lagrange multipliers, and W is a $p \times |B|$ matrix whose columns are the gradients of the binding constraints $\nabla c_j, j \in B$.

As the sample size $N/2K \to \infty$, $Y \to a\nabla\eta$ w.p. 1, and then the solution of (75) and (76) also converges to $a\nabla\eta$ w.p. 1. Therefore, if at the current solution $\bar{\zeta}^*$: (i) $\nabla\eta$ can be expressed as a linear combination of $\nabla c_j, j \in B$ (Condition 7); (ii) all the coefficients in this linear combination are positive (Assumption 5); and (iii) the sample size is large enough; then, all the Lagrange multipliers in the solution of (75) and (76) are strictly positive. Therefore, the cone $\mathcal{C}$ can be replaced by a subspace $\mathcal{L}$ without changing the solutions corresponding to the minimal values d_1 and d_2. In other words, the unconstrained quadratic programming problems

$$\hat{d}_1 \equiv \min_{\lambda \in \mathbb{R}^{|B|}} (Y - W\lambda)^T \hat{\Sigma}^{-1} (Y - W\lambda) \tag{77}$$

and

$$\hat{d}_2 \equiv \min_{\lambda \in \mathbb{R}^{|B|}} (Y - W\lambda)^T (Y - W\lambda) \tag{78}$$

have the same solutions as the constrained problems (75) and (76), respectively.[13] A key advantage of the formulations defined by Equations (77) and (78) over the corresponding formulations defined by Equations (75) and (76) is that the former can be perceived as projections of the vector Y to the subspace that is defined by the columns of the matrix W, and, therefore, the corresponding optimal solutions can be expressed in closed-form through the employment of appropriate projection matrices.

[13] On the other hand, if the formulations of Equations (77) and (78) do not have the same optimal solutions with their respective formulations of Equations (75) and (76), then the optimal solution of Equation (75) or (76) should have at least one zero coefficient, and Assumption 5 will be violated.

A hypothesis test using the distance $\hat{d}_1$ The optimal solution of the formulation of Equation (77) is

$$(W^T\hat{\Sigma}^{-1}W)^{-1}W^T\hat{\Sigma}^{-1}Y$$

Let $Q_1 \equiv W(W^T\hat{\Sigma}^{-1}W)^{-1}W^T\hat{\Sigma}^{-1}$. Then the test statistic $\hat{d}_1$ can be expressed as

$$\hat{d}_1 = Y^T(I - Q_1)^T\hat{\Sigma}^{-1}(I - Q_1)Y \tag{79}$$

where I is the identity matrix with the proper dimension.

If the null hypothesis is true, i.e., $\mathbb{E}[Y] \in \mathcal{C}$, then $\hat{d}_1$ is central-chi-squared distributed with $p - |B|$ degrees of freedom. This conclusion can be proved from the following result in [69] (c.f. Corollary 5.1.3a, pg. 201):

Proposition 9 *Consider a multi-normally distributed random vector Y with mean μ and positive definite covariance matrix Σ, and also let A be a symmetric matrix. Then, a set of necessary and sufficient conditions for Y^TAT to be a non-central chi-square random variable with ν degrees of freedom and non-centrality parameter $\delta^2 = \mu^TA\mu$ is as follows:*

(i). $\texttt{trace}(A\Sigma) = \nu$.

(ii). $A\Sigma A = A$.

Before checking the conditions of Proposition 9, let us get some more properties of the matrix Q_1. First, it can be checked that

$$Q_1^T\hat{\Sigma}^{-1}Q_1 = \hat{\Sigma}^{-1}Q_1 = Q_1^T\hat{\Sigma}^{-1} \tag{80}$$

More specifically,

$$\begin{aligned}
Q_1^T\hat{\Sigma}^{-1}Q_1 &= \left(W(W^T\hat{\Sigma}^{-1}W)^{-1}W^T\hat{\Sigma}^{-1}\right)^T \cdot \hat{\Sigma}^{-1} \cdot W(W^T\hat{\Sigma}^{-1}W)^{-1}W^T\hat{\Sigma}^{-1} \\
&= \hat{\Sigma}^{-1}W(W^T\hat{\Sigma}^{-1}W)^{-1}W^T\hat{\Sigma}^{-1}W(W^T\hat{\Sigma}^{-1}W)^{-1}W^T\hat{\Sigma}^{-1} \\
&= \hat{\Sigma}^{-1}W(W^T\hat{\Sigma}^{-1}W)^{-1}\left(W^T\hat{\Sigma}^{-1}W(W^T\hat{\Sigma}^{-1}W)^{-1}\right)W^T\hat{\Sigma}^{-1} \\
&= \hat{\Sigma}^{-1}W(W^T\hat{\Sigma}^{-1}W)^{-1}W^T\hat{\Sigma}^{-1} \\
&= \hat{\Sigma}^{-1}\left(W(W^T\hat{\Sigma}^{-1}W)^{-1}W^T\hat{\Sigma}^{-1}\right) \\
&= \hat{\Sigma}^{-1}Q_1
\end{aligned}$$

and

$$\begin{aligned}
\hat{\Sigma}^{-1}Q_1 &= \hat{\Sigma}^{-1}\left(W(W^T\hat{\Sigma}^{-1}W)^{-1}W^T\hat{\Sigma}^{-1}\right) \\
&= \left(\hat{\Sigma}^{-1}W(W^T\hat{\Sigma}^{-1}W)^{-1}W^T\right)\hat{\Sigma}^{-1} \\
&= \left(W(W^T\hat{\Sigma}^{-1}W)^{-1}W^T\hat{\Sigma}^{-1}\right)^T\hat{\Sigma}^{-1} \\
&= Q_1^T\hat{\Sigma}^{-1}
\end{aligned}$$

A further result implied by Equation (80) is that

$$(I - Q_1)^T\hat{\Sigma}^{-1}(I - Q_1) = \hat{\Sigma}^{-1}(I - Q_1) = (I - Q_1)^T\hat{\Sigma}^{-1} \tag{81}$$

In the context of the test statistic $\hat{d}_1$ expressed in Equation (79), Y is approximately multi-normally distributed with mean $a\nabla\eta$ and covariance $\hat{\Sigma}$. Also, $A = (I - Q_1)^T\hat{\Sigma}^{-1}(I - Q_1)$, and it can be equivalently expressed in the two other forms in (81). Using these alternative expressions of A, condition (ii) in Proposition 9 can be validated as follows:

$$A\Sigma A = (I - Q_1)^T\hat{\Sigma}^{-1}\hat{\Sigma}\hat{\Sigma}^{-1}(I - Q_1) = (I - Q_1)^T\hat{\Sigma}^{-1}(I - Q_1) = A$$

Another property of Q_1 that results from its nature as a projection operator is that the trace of Q_1 is equal to its rank, which is the number of binding constraints

$|B|$ [70]. Then, the trace of the matrix $A\Sigma$ is:

$$\texttt{trace}(A\Sigma) = \texttt{trace}((I - Q_1)^T\hat{\Sigma}^{-1}\hat{\Sigma}) = \texttt{trace}(I - Q_1) = p - \texttt{trace}(Q_1) = p - |B|$$

Finally, under the null hypothesis considered in this part of the discussion, the non-centrality parameter δ^2 of Proposition 9 can be shown to be equal to zero, which, when combined with the above results, implies that $\hat{d}_1$ is central-chi-squared distributed with $\texttt{trace}(A\Sigma) = p - |B|$ degrees of freedom. Indeed, under the null hypothesis, $\mathbb{E}[Y] = a\nabla\eta = Wv$ for some vector v, and therefore,

$$\begin{aligned}
\delta^2 &= v^T W^T \cdot \hat{\Sigma}^{-1}(I - Q_1) \cdot Wv \\
&= v^T W^T \hat{\Sigma}^{-1} Wv - v^T W^T \hat{\Sigma}^{-1} \left(W(W^T\hat{\Sigma}^{-1}W)^{-1} W^T \hat{\Sigma}^{-1} \right) Wv \\
&= v^T W^T \hat{\Sigma}^{-1} Wv - v^T W^T \hat{\Sigma}^{-1} W \left((W^T\hat{\Sigma}^{-1}W)^{-1} W^T \hat{\Sigma}^{-1} W \right) v \\
&= v^T W^T \hat{\Sigma}^{-1} Wv - v^T W^T \hat{\Sigma}^{-1} Wv \\
&= 0
\end{aligned}$$

In conclusion, the above discussion has established that for a sufficiently large sample size $N/2K$ (or sufficiently large N for a fixed K), and under the assumption that the null hypothesis $\nabla\eta \in \mathcal{C}$ holds, the distance $\hat{d}_1$ follows a chi-square distribution with $p - |B|$ degrees of freedom. Therefore, it can function as a test statistic for the null hypothesis with p-value

$$\mathcal{P}_1 \equiv \mathbb{P}\left\{ \chi^2_{p-|B|} \geq \hat{d}_1 \right\} \tag{82}$$

However, $\hat{d}_1$ involves the computation of the inverse matrix $\hat{\Sigma}^{-1}$, an operation of rather high complexity and potential numerical instability. This last remark motivates the development of a second hypothesis test that is based on the distance $\hat{d}_2$.

An alternative termination hypothesis test using the distance $\hat{d}_2$ The optimal solution for the formulation of Equation (78) is $(W^TW)^{-1}W^TY$. Let $Q_2 \equiv$

$W(W^TW)^{-1}W^T$. Then, the test statistic defined by Equation (78) is

$$\hat{d}_2 = Y^T(I - Q_2)Y \tag{83}$$

The computation of $\hat{d}_2$ does not involve any matrix inversion, but the distribution of $\hat{d}_2$ is no longer chi-squared. However, from [69], if a random vector Y is multi-normally distributed with mean μ and positive definite covariance matrix Σ, then, for any given symmetric matrix A, the quadratic form Y^TAY can be written as

$$Y^TAY = \sum_{j=1}^{p} \beta_j (U_j + b_j)^2 \tag{84}$$

$$= \sum_{j=1}^{p} \beta_j U_j^2 + 2\sum_{j=1}^{p} b_j U_j + \sum_{j=1}^{p} b_j^2 \tag{85}$$

In Equations (84) and (85), $\beta_1, \ldots, \beta_p$ are the eigenvalues of the matrix $\Sigma^{\frac{1}{2}}A\Sigma^{\frac{1}{2}}$, and $U_1, \ldots, U_j$ are independent standard normally distributed random variables. Furthermore, the constant term $\sum_{j=1}^{p} b_j^2 = \mu^T A\mu$. Therefore, all the b_j's are zero if and only if $\mu^T A\mu = 0$.

If the null hypothesis holds, then we can prove $\mathbb{E}[Y]^T(I - Q_2)\mathbb{E}[Y] = 0$, in a way similar to that we used for proving $\delta^2 = 0$ in the development of the hypothesis test for $\hat{d}_1$. Therefore, the distribution of $\hat{d}_2$ is the weighted sum of independent central chi-squared distributed random variables with one degree of freedom:

$$\hat{d}_2 = \sum_{j=1}^{p} \beta_j U_j^2 \tag{86}$$

And the weights β_j's are eigenvalues of $\Sigma^{\frac{1}{2}}(I - Q_2)\Sigma^{\frac{1}{2}}$, or equivalently, eigenvalues of $(I - Q_2)\Sigma$.[14]

The distribution of $\hat{d}_2$ can be approximated with a chi-squared distribution using Pearson's approach in [69] (c.f. pages 164-165). More specifically, for any positive

[14]This equivalence comes from the notion of *similarity* between matrices. More specifically, for any square matrices A and B, if there exists an invertible matrix P such that $B = P^{-1}AP$, then the matrices A and B are called *similar*, and they have the same eigenvalues [70].

integer k, let

$$\begin{aligned}\omega(k) &\equiv \sum_{j=1}^{p} \beta_j^k \\ &= \texttt{trace}\left(\hat{\Sigma}^k\right)\end{aligned} \tag{87}$$

and

$$\begin{aligned}\nu &= \omega(2)^3/\omega(3)^2 \\ a_0 &= \omega(1) - \omega(2)^2/\omega(3) \\ a_1 &= \omega(3)/\omega(2)\end{aligned} \tag{88}$$

Then,

$$\sum_{j=1}^{p} \alpha_j U_j^2 \approx a_1 \chi_\nu^2 + a_0 \tag{89}$$

where χ_ν^2 is a chi-squared distributed random variable with ν degrees of freedom.

Finally, the p-value for this hypothesis test is:

$$\mathcal{P}_2 \equiv \mathbb{P}\left\{\chi_\nu^2 \geq \frac{\hat{d}_2 - a_0}{a_1}\right\} \tag{90}$$

□

Some practical considerations Both of the test statistics that we presented in the previous paragraphs will provide us with a p-value. However, next we discuss some more practical aspects for the terminating condition that is eventually employed in the proposed SA algorithm. These aspects intend to strengthen the discerning power of the original tests, and also address some cases of a more pathological nature.

More specifically, for a given confidence level $100(1-\alpha)\%$, besides the rejection ($p < \alpha$) and non-rejection ($p \geq \alpha$) of the null hypothesis, there are two additional possible outcomes that can arise during the execution of these tests:

1. The number of binding constraints equals to the number of decision variables. In this case, both hypothesis tests involve a projection on the single point that

represents the intersection of all the binding constraints, and the expression $p-|B|$ that defines the degrees of freedom involved will be equal to zero. Hence, these hypothesis tests are not functional. However, the aforementioned points can be recognized as "corner" – or more formally, extreme – points for the underlying solution space, and persistence of the algorithm to remain (or return) to these points can be taken as a signal of the optimality of these points, and a reason for the termination of the algorithm.

2. Some coefficients of the projection of Y are not positive, i.e., $Q_1 \cdot Y \not> \mathbf{0}$ for the $\hat{d}_1$-test or $Q_2 \cdot Y \not> \mathbf{0}$ for the $\hat{d}_2$-test. If the strict complementarity assumption for the solution space holds, then this outcome implies either that the sample size is not large enough or that the null hypothesis does not hold. Since these two reasons are not distinguishable, the corresponding test is inconclusive about potential satisfaction of the KKT optimality conditions by the considered solution.

In view of all the above remarks, the eventual implementation of the tests for the termination condition that were presented in this section, involves two counters i_1, i_2 and two thresholds n_1, n_2. Counter i_1 keeps track of the number of the consecutive non-rejections by the applied hypothesis test, while counter i_2 keeps track of the number of the consecutive visits of the algorithm at some corner point. More specifically, if a visit to a corner point takes place, then counter i_1 remains unchanged, and counter i_2 increases by one; if $Q_{1,2} \cdot Y \not> \mathbf{0}$ during the execution of the corresponding test, then both counters remain unchanged; if a rejection of the null hypothesis occurs, then both i_1 and i_2 are reset to zero; finally, if the test simply fails to reject the corresponding null hypothesis, then i_1 increases by one, and i_2 is reset to zero. The algorithm terminates if any of the counters $i_i,\ i = 1, 2$, reaches the corresponding limit n_i. The values of n_1 and n_2 should be carefully designed for each test.

Furthermore, there is an overall upper bound $n_{\max}$ for the total number of iterations that are executed by the algorithm. This upper bound is imposed by the practical limitation on the available computational resources. Hence, after completing $n_{\max}$ iterations, the algorithm will terminate even if the hypothesis test indicates that the algorithm can make further progress.

5.3.4 The practical version of the proposed SA algorithm

In this section we present the complete SA algorithm that integrates the statistical enhancements that were introduced in the previous two sections, and we also present some numerical results indicating the efficacy of these enhancements.

Compared to the implementation of the basic and asymptotically convergent SA algorithm in Section 5.2.4, this new algorithm has some different input. The step size is not a function of the iteration number n but a constant γ. The sample size is also not a function of n, but it is determined in an adaptive way that also employs as input the initial number of the regenerative cycles N_0 (that must be divisible by $2K$). Two additional inputs are the parameters α_1 and α_2 that define, respectively, the confidence levels for the sample-size control and the termination hypothesis test. The parameter $t_{\max}$ is used to set a hard limit on the sample size, so that the simulation in one iteration can terminate earlier in case that the estimated N_{new} is too large.[15] The parameters $n_{\max}$, n_1 and n_2 are those introduced in the concluding part of Section 5.3.3. Finally, the questionmark at Line 15 of the presented algorithm implies that the value returned by the expression $N_{\text{new}} > N/2K$ is of Boolean type.

The significance and the efficacy of the algorithmic enhancements that were presented in the previous sections are assessed against the performance of the baseline algorithm in Section 5.3.1, through a numerical experiment that employs the same 16 CRL configurations used in the baseline experiment of that section. In fact, this

[15] But the constraint $t < t_{\max}$ is not integrated in the loop condition at Line 9 of Algorithm 6, so that the simulation always samples complete groups of $2K$ regenerative cycles.

Algorithm 6 A practical implementation of the stochastic approximation algorithm for the considered optimization problem

Input: RAS Φ, δ, ξ, $trial$, γ, K, N_0, $n_{\max}$, $t_{\max}$, α_1, l_0, α_2, n_1, n_2.
Output: GSPN, DAP, Ξ, $\bar{\zeta}$, $idx^S(\cdot)$.

1: Model RAS Φ as GSPN with the methods in Section 3.1.
2: Solve for the DAP of Φ with the methods in Section 2.2 and code the DAP and the non-deliberately-idling constraint into GSPN as Π_2. The finally adopted policy space should be $\hat{\Pi}_3^S$.
3: **for** $n = 1 \to n_{\max}$ **do**
4: Starting from the initial marking M_0, simulate the GSPN for $trial$ non-self-loop transitions to get the most visited tangible marking M^*. Also extend Ξ, $\bar{\zeta}$ and $idx^S(\cdot)$ with any newly encountered random switches, and initialize the new components of $\bar{\zeta}$ according to ξ.
5: $t \leftarrow 0$; $x \leftarrow \texttt{odd}$; $N \leftarrow N_0$; $i \leftarrow 1$; $sampleMore \leftarrow$ **true**.
6: **while** $sampleMore \wedge t < t_{\max}$ **do**
7: **while** $i < N/2K \wedge t < t_{\max}$ **do**
8: Set the 8 counters for the respective 8 sums in Eq. (46) to 0 or $\mathbf{0}$; Set the single samples $\hat{z}(\omega, \mathbf{1})$, $\widehat{\nabla z}(\omega, \mathbf{1})$, $\hat{z}(\omega, \hat{r})$ and $\widehat{\nabla z}(\omega, \hat{r})$ to 0 or $\mathbf{0}$; $pair \leftarrow 0$.
9: **while** $pair < K$ **do**
10: Implement the same block as Lines 9–19 of Algorithm 5, for calculations of the single samples $\hat{z}(\omega, \mathbf{1})$, $\widehat{\nabla z}(\omega, \mathbf{1})$, $\hat{z}(\omega, \hat{r})$ and $\widehat{\nabla z}(\omega, \hat{r})$, and the updates the counters for the 8 sums in Eq. (46). Also update the transition counter t and the regenerative cycle pair counter $pair$.
11: **end while**
12: Compute $\tilde{Y}_i$ using Eq. (46); Update Y_{A} and Y_{Sq} using Eqs. (55) and (57); $i \leftarrow i + 1$.
13: **end while**
14: Compute $\hat{\Sigma}$ using Eq. (58); Compute N_{new} using Eq. (65), (72) or (74).
15: $sampleMore \leftarrow (N_{\mathrm{new}} > N/2K?)$; $N \leftarrow 2KN_{\mathrm{new}}$.
16: **end while**
17: Test for termination condition and update the counters i_1, i_2 as Section 5.3.3.
18: **if** $i_1 \geq n_1 \vee i_2 \geq n_2$ **then**
19: **return** GSPN, DAP, Ξ, $\bar{\zeta}$, $idx^S(\cdot)$.
20: **end if**
21: Compute $\widehat{\nabla \eta}$ using Eq. (48) with $\tau_N = t$.
22: $\bar{\zeta} \leftarrow \bar{\zeta} + \gamma \cdot \widehat{\nabla \eta}$.
23: Project $\bar{\zeta}$ with respect to every random switch onto the feasible region defined by Eqs. (34) and (35), using Algorithm 7 in Appendix D.
24: **end for**
25: **return** GSPN, DAP, Ξ, $\bar{\zeta}$, $idx^S(\cdot)$.

new experiment has the same parameter settings as the baseline experiment, with some differences that can be described as follows: $n_{\max} = 500$, but it has the modified meaning that the executed iterations may be either 500 or smaller, according to the results of the employed termination hypothesis test. The number of the regenerative cycles sampled at each iteration is no longer fixed to $N = 10,000$. Instead, it is determined according to the methodology described in Section 5.3.2. The parameters K and α_1 that are employed by that methodology are set to the values $K = 100$ and $\alpha_1 = 0.2$, which are the same values as those used in the experiment in Section 5.3.2. On the other hand, the initial number of regenerative cycles N_0 is set to $6,000$, in order to control further the effort expended on this preliminary sampling phase. In a similar spirit, l_0 is set to $l_0 = 0.02$ to relax a little more the precision requirement on the sample size. On the other hand, the ratio $N_0/2K$ implies 30 samples of $\tilde{Y}_i$ during the initial phase of the aforementioned methodology, and this number is still sufficiently large to ensure the normal approximation of Y_{A}. With regards to the options that are provided by Algorithm 6 about the implementation of the various alternative statistical inference methods, in the presented experiment, we employed Equation (74) in Line 14 of the algorithm, and the hypothesis test based on the distance $\hat{d}_2$ was selected for Line 17. Finally, some other parameters were set as follows: $t_{\max} = 5 \times 10^7$, $\alpha_2 = 0.7$, $n_1 = 4$, and $n_2 = 3$.

The layout of Table 5.4 is the same as that of Table 5.1 with some slight differences: an extra column n_{stop} reports the iteration number when the algorithm terminated, either because of the hypothesis testing in Section 5.3.3 or by reaching the upper bound $n_{\max}$; furthermore, since not all the instances stop at $n_{\max} = 500$, but they may stop at some $n_{\mathrm{stop}} < 500$, the column $\eta(\bar{\zeta}_{500})$ is replaced by the column $\eta(\bar{\zeta}_{n_{\mathrm{stop}}})$.

We can see in Table 5.4 that the solutions returned by Algorithm 6 have almost the same objective values; this result is indicated even more emphatically in Figure 5.5. On the other hand, thanks to the enhancements implemented in Algorithm 6, the

Table 5.4: The values of the objective function at the solution points returned by Algorithm 6 for the first 16 CRL configurations of Table 4.1

Conf.	Objective values				n_{stop}	Trans. ($\times 10^6$)	Time (sec)
	Max	$\eta(\bar{\zeta}_0)$	$\eta(\bar{\zeta}_{100})$	$\eta(\bar{\zeta}_{n_{\text{stop}}})$			
1	0.48000	0.47333	0.47988	0.47993	131	5.65	11
2	0.44444	0.43478	N/A	0.44324	28	0.88	2
3	0.49254	0.48495	0.48933	0.48940	111	14.33	42
4	0.49959	0.49811	0.49872	0.49883	500	98.94	241
5	0.50000	0.50000	0.49999	0.49999	500	240.06	593
6	0.46411	0.45251	N/A	0.46208	46	3.82	7
7	0.49310	0.48348	0.48623	0.48658	500	187.50	400
8	0.49820	0.49225	0.49441	0.49541	500	240.26	559
9	0.49999	0.49969	0.49979	0.49968	500	459.71	1,256
10	0.32234	0.30649	0.30794	0.31139	500	1891.05	5,500
11	0.43734	0.43359	0.43505	0.43541	361	682.27	1,958
12	0.42225	0.41539	0.42034	0.42114	500	351.84	1,101
13	0.43212	0.41808	0.42365	0.42425	372	492.71	1,381
14	0.41063	0.39231	0.40055	0.40293	479	230.87	626
15	0.37667	0.37359	0.37453	0.37539	469	1596.22	5,438
16	0.35729	0.35453	0.35550	0.35638	263	491.15	1,526

total number of executed transitions presents an average reduction by 43.10%, and the required computing times are reduced by 40.64%. The reduction of the Markovian transitions and computing time for each configuration are depicted in Figure 5.6.

In summary, the presented results indicate that alterations that were performed on the baseline SA algorithm with respect to the determination of the necessary sampling and the terminating condition, have led to considerable reductions of the computational cost involved in the algorithm execution on any given RAS configuration, and they have maintained the same level of optimality of the returned solutions. However, these alterations have also introduced some new control parameters in the algorithm logic, and the values of these parameters should be carefully determined for a successful realization of the aforementioned enhancements.

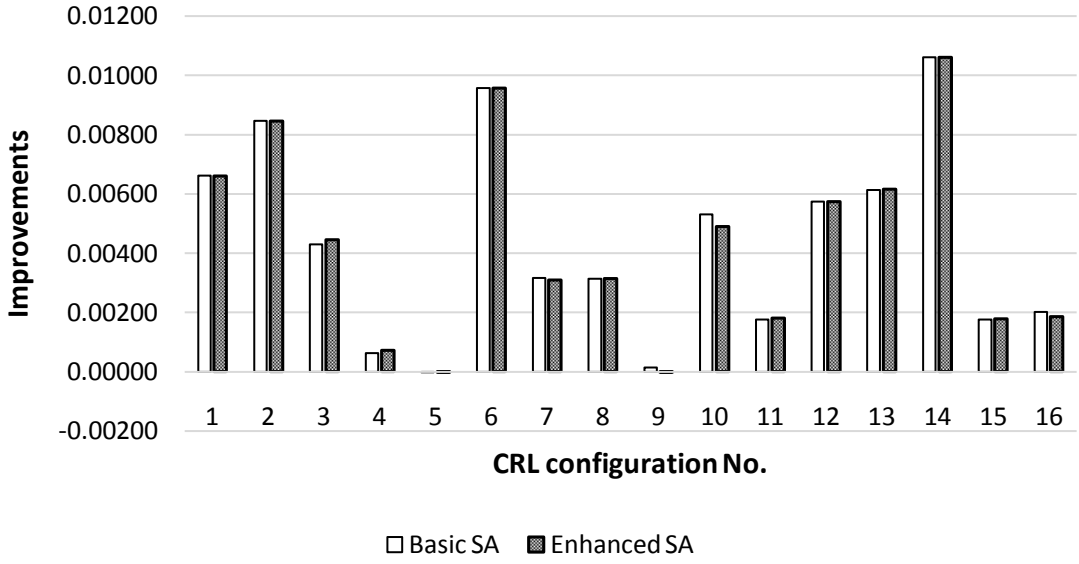


Figure 5.5: Comparison of the throughput improvements obtained from the basic and the enhanced SA algorithms for the 16 CRL configurations of Table 4.1

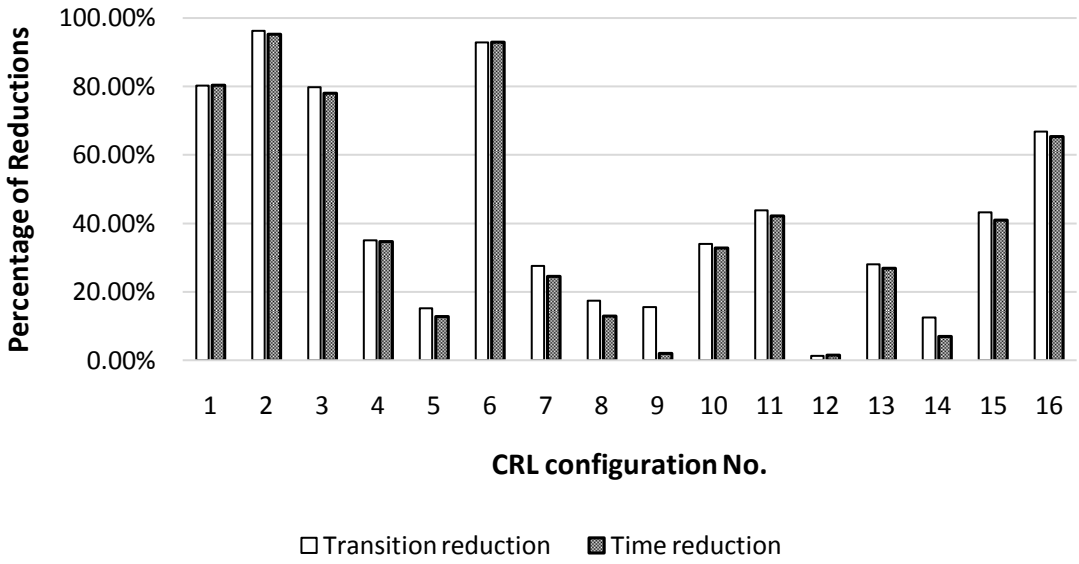


Figure 5.6: The reductions with respect to the total number of transitions and the computing time incurred by the presented algorithmic enhancements during the algorithm application on the 16 CRL configurations of Table 4.1

5.4 Extensions for partially disaggregated random switches

The first part of this section discusses the necessary extensions for the SA algorithm in order to cope with random switches that are partially disaggregated under the representation of Equation (29) in Section 4.3. Subsequently, a numerical experiment is performed to demonstrate the potential improvement that can be brought by partial disaggregation.

5.4.1 Modifying the SA algorithm to cope with partial disaggregation

The introduction of partially disaggregated random switches under the representation of Equation (29) in Section 4.3 does not alter the fundamental structure and logic of the presented SA algorithms, but it impacts the computation of the one-step transition probabilities for the underlying DTMC $\hat{\mathcal{M}}$ and their gradients with respect to the decision variables $\bar{\zeta}$. Also, from the discussion of Section 4.3, it should be evident that the decision variables that correspond to this class of random switches are totally free variables. Hence, there is no need for the projection operation of Appendix D for this class of random switches.[16] Next, we will derive the changes in the computation of the one-step transition probabilities and their gradients that are necessitated by the introduction of partially disaggregated random switches.

As mentioned above, the introduction of partially disaggregated random switches leaves unchanged the basic scheme for the computation of the one-step transition probabilities from one tangible marking $M_{\mathcal{T}}$ of the DTMC $\hat{\mathcal{M}}$ to another – i.e., the elements $P[M_{\mathcal{T}}, M'_{\mathcal{T}}]$ in the transition probability matrix of the DTMC $\hat{\mathcal{M}}$ – and of their gradients. More specifically, the computation of the elements $P[M_{\mathcal{T}}, M'_{\mathcal{T}}]$

[16] Returning to the content of Footnote 8 in Section 4.3, we notice that the explicit enforcement of the randomization that is implied by the constraints (15) and (16) of formulation (14)–(16), would necessitate the introduction of one such set of constraints for every marking that is covered by the disaggregated random switches, and this can easily be an intractable proposition even for moderately small RAS configurations. In particular, the projection of the vectors of the relevant decision variables to the subspaces defined by these constraints would be computationally intractable.

is still based on the double summation of Equation (49) in Section 5.2.2. And the gradient of these probabilities is still characterized by the sum of Equation (50). Furthermore, for any untimed transition firing sequence σ leading from a marking $\hat{M}$ that results from the firing of a timed transition t to a tangible marking $M'_{\mathcal{T}}$, the inductive expressions (51) and (52) that can compute $x_{\hat{M},\sigma}$ and $\nabla x_{\hat{M},\sigma}$ do not change as well. The only necessary change in the case that the enabling pattern $\mathcal{E} = \mathcal{E}_u^{\hat{\Pi}_2}(\hat{M}_i)$ corresponds to a partially disaggregated type, is for the expressions (53) and (54) that provide respectively the selection probability $\hat{p}_i$ and the partial derivatives $\frac{\partial \hat{p}_i}{\partial \bar{\zeta}[\hat{k}]}$. More specifically, the selection probability $Z_{\hat{M}_i}[\hat{t}_i]$ of Equation (53) must be replaced by

$$\hat{p}_i = \frac{\exp\Big(\sum_{k=1}^{K} \psi_k(\hat{M}_i) \cdot \bar{\zeta}\big[idx^{DA}(\mathcal{E}, \hat{t}_i)[k]\big]\Big)}{\sum_{t \in \mathcal{E}} \exp\Big(\sum_{k=1}^{K} \psi_k(\hat{M}_i) \cdot \bar{\zeta}\big[idx^{DA}(\mathcal{E}, t)[k]\big]\Big)} \tag{91}$$

Let

$$\begin{aligned} A(M,t) &\equiv \exp\Big(\sum_{k=1}^{K} \psi_k(M) \cdot \bar{\zeta}\big[idx^{DA}(\mathcal{E}_u^{\hat{\Pi}_2}(M), t)[k]\big]\Big) \\ B(M) &\equiv \sum_{t \in \mathcal{E}_u^{\hat{\Pi}_2}(M)} A(M,t) \end{aligned} \tag{92}$$

Then, Equation (91) can be expressed more compactly as

$$\hat{p}_i = \frac{A(\hat{M}_i, \hat{t}_i)}{B(\hat{M}_i)} \tag{93}$$

For the calculation of the corresponding partial derivatives, first we notice that if the considered decision variable $\bar{\zeta}[\hat{k}]$ is not related to this random switch, i.e., $\nexists \hat{t}, \tilde{k}$ such that $idx^{DA}(\mathcal{E}, \hat{t})[\tilde{k}] = \hat{k}$, then $\frac{\partial \hat{p}_i}{\partial \bar{\zeta}[\hat{k}]} = 0$. Otherwise, $\bar{\zeta}[\hat{k}]$ is related to this random switch, and suppose that $idx^{DA}(\mathcal{E}, \hat{t})[\tilde{k}] = \hat{k}$. Then, the formulae that provide the corresponding partial derivatives are different depending on whether $\hat{t}$ is fired at $\hat{M}_i$.

Using the relevant notation of Section 5.2.2, the easier case is that where $\hat{t} \neq \hat{t}_i$. Then, the term $\exp\Big(\psi_{\tilde{k}}(\hat{M}_i)\bar{\zeta}[\hat{k}]\Big)$ appears only in the denominator of Equation (93). Let

$$\hat{x} \equiv \exp\Big(\psi_{\tilde{k}}(\hat{M}_i)\bar{\zeta}[\hat{k}]\Big) \tag{94}$$

Then,

$$\begin{aligned}
\frac{\partial \hat{p}_i}{\partial \bar{\zeta}[\hat{k}]} &= \frac{\partial}{\partial \bar{\zeta}[\hat{k}]} \left(\frac{A(\hat{M}_i, \hat{t}_i)}{\Big(B(\hat{M}_i) - A(\hat{M}_i, \hat{t})\Big) + \frac{A(\hat{M}_i, \hat{t})}{\hat{x}} \cdot \hat{x}} \right) \\
&= \frac{\mathrm{d}}{\mathrm{d}\hat{x}} \left(\frac{A(\hat{M}_i, \hat{t}_i)}{\Big(B(\hat{M}_i) - A(\hat{M}_i, \hat{t})\Big) + \frac{A(\hat{M}_i, \hat{t})}{\hat{x}} \cdot \hat{x}} \right) \cdot \frac{\partial \hat{x}}{\partial \bar{\zeta}[\hat{k}]} \\
&= \frac{-A(\hat{M}_i, \hat{t}_i) \cdot \frac{A(\hat{M}_i, \hat{t})}{\hat{x}}}{\Big(B(\hat{M}_i)\Big)^2} \cdot \Big(\psi_{\tilde{k}}(\hat{M}_i) \cdot \hat{x}\Big) \\
&= \left(\frac{A(\hat{M}_i, \hat{t}_i)}{B(\hat{M}_i)} \right) \cdot \left(\frac{-\psi_{\tilde{k}}(\hat{M}_i) A(\hat{M}_i, \hat{t})}{B(\hat{M}_i)} \right) \\
&= \hat{p}_i \cdot \left(\frac{-\psi_{\tilde{k}}(\hat{M}_i) A(\hat{M}_i, \hat{t})}{B(\hat{M}_i)} \right)
\end{aligned} \tag{95}$$

The computation of the derivative with respect to $\hat{x}$ in the third step above is based on the fact that the items $A(\hat{M}_i, \hat{t}_i)$, $\Big(B(\hat{M}_i) - A(\hat{M}_i, \hat{t})\Big)$, and $\frac{A(\hat{M}_i, \hat{t})}{\hat{x}}$ are independent from $\hat{x}$.

Next, we consider the case where $\hat{t} = \hat{t}_i$. Then, with $\hat{x}$ still defined as in (94), the sought partial derivative is computed as follows:

$$\begin{aligned}
\frac{\partial \hat{p}_i}{\partial \bar{\zeta}[\hat{k}]} &= \frac{\partial}{\partial \bar{\zeta}[\hat{k}]} \left(\frac{\frac{A(\hat{M}_i, \hat{t})}{\hat{x}} \cdot \hat{x}}{\Big(B(\hat{M}_i) - A(\hat{M}_i, \hat{t})\Big) + \frac{A(\hat{M}_i, \hat{t})}{\hat{x}} \cdot \hat{x}} \right) \\
&= \frac{\mathrm{d}}{\mathrm{d}\hat{x}} \left(\frac{\frac{A(\hat{M}_i, \hat{t})}{\hat{x}} \cdot \hat{x}}{\Big(B(\hat{M}_i) - A(\hat{M}_i, \hat{t})\Big) + \frac{A(\hat{M}_i, \hat{t})}{\hat{x}} \cdot \hat{x}} \right) \cdot \frac{\partial \hat{x}}{\partial \bar{\zeta}[\hat{k}]} \\
&= \frac{\frac{A(\hat{M}_i, \hat{t})}{\hat{x}} \cdot B(\hat{M}_i) - A(\hat{M}_i, \hat{t}) \cdot \frac{A(\hat{M}_i, \hat{t})}{\hat{x}}}{\Big(B(\hat{M}_i)\Big)^2} \cdot \Big(\psi_{\tilde{k}}(\hat{M}_i) \cdot \hat{x}\Big) \\
&= \hat{p}_i \cdot \psi_{\tilde{k}}(\hat{M}_i) \left(1 - \frac{A(\hat{M}_i, \hat{t})}{B(\hat{M}_i)} \right)
\end{aligned} \tag{96}$$

As a closing remark, we notice that Equations (95) and (96), when combined with Equation (92), imply that the corresponding partial derivatives $\frac{\partial \hat{p}_i}{\partial \bar{\zeta}[\hat{k}]}$ are well defined and their absolute values are bounded by the absolute values of the corresponding parameters $\psi_{\tilde{k}}(\hat{M}_i)$. This bounding further implies the bounding of the vectors $\widehat{\nabla \eta}$

that will be employed by the corresponding SA algorithm. On the other hand, we cannot establish a bound for the corresponding decision variables $\bar{\zeta}$,[17] but it is also true that we have not encountered any numerical instability with respect to these variables in our numerical experimentation.[18]

5.4.2 Demonstrating the improvement potential of partial disaggregation

In this section we illustrate the potential performance improvements that can be brought about by the proposed partial disaggregation of the static random switches. The presented experiment employs the CRL configuration #13 in Table 4.1. This configuration was selected because (i) it has a manageable number of random switches for the needs of the pursued demonstration, and (ii) the results of Table 5.4 indicate significant potential for performance improvement through the employed relaxation (c.f., the two columns "Max" and "$\eta(\bar{\zeta}_{n_{\text{stop}}})$" in that table).

More specifically, the considered CRL configuration consists of $m = 3$ workstations and $n = 5$ processing stages. The route that is defined by these processing stages through the line workstations (WS), is $WS_1 \rightarrow WS_2 \rightarrow WS_1 \rightarrow WS_3 \rightarrow WS_2$. If the corresponding GSPN is constructed according to the method presented in Section 3.2, [19] then the random switches of the underlying $\hat{\Pi}_2$-conditional state space can be categorized into four enabling patterns, or four static random switches; these four enabling patterns are denoted as $\mathcal{E}_1 \equiv \{t_0, t_6\}$, $\mathcal{E}_2 \equiv \{t_3, t_{12}\}$, $\mathcal{E}_3 \equiv \{t_0, t_5\}$ and $\mathcal{E}_4 \equiv \{t_2, t_{11}\}$. $\mathcal{E}_1$ and $\mathcal{E}_3$ model the conflicts at workstation 1 between processing stages 1 and 3. Among them, $\mathcal{E}_1$ models the conflict with respect to the server

[17] One way to explain the inability to bound the decision variables $\bar{\zeta}$ that are employed by the proposed partial disaggregation scheme, is by noticing that the translation of the variable vector $\bar{\zeta}$ that corresponds to a disaggregated static random switch by a certain constant α will retain the values for the selection probabilities $\hat{p}_i$ that are defined by Equation (91).

[18] A possible stabilizing factor for the values of the decision variables $\bar{\zeta}$ might come from the fact that they need to establish efficient distributions for all the markings M that correspond to the disaggregated static random switch, and this requirement implies some "antithetic" trends in the corresponding pricing.

[19] More specifically, the numbering of the places and the transitions of this GSPN model parallels the corresponding logic that was introduced in Table 3.1 of Example 1.

allocation, and $\mathcal{E}_3$ models the conflict with respect to the buffer allocation. $\mathcal{E}_2$ and $\mathcal{E}_4$ model the conflicts at workstation 2 between processing stages 2 and 5, respectively for the server and the buffer allocation.

The feature functions $\psi_k(\cdot)$ that we considered in this experiment are the components of the submarking that corresponds to the process places of the underlying GSPN. Since there are $3 \times 5 - 2 = 13$ such process places in the considered GSPN, and two enabled transitions in each of the four enabling patterns $\mathcal{E}_1$ – $\mathcal{E}_4$ mentioned above, each partially disaggregated random switch will employ $13 \times 2 = 26$ free decision variables. This, in turn, implies that the scheduling formulation that is obtained by substituting (only) one of the four static random switches by a partially disaggregated random switch, will have $4 - 1 + 26 = 29$ decision variables; 3 corresponding directly to selection probabilities for the three employed static random switches, and 26 free variables will define the selection probability distribution for the fourth random switch, according to the logic of Equation (29). The above discussion also renders clear that, for a fixed set of features to be used in the considered disaggregation process, the increase in the number of the decision variables that will result from the disaggregation of N static random switches is only $O(N)$.

We ran Algorithm 6 five times on the considered CRL configuration, under the following five policy spaces: $\hat{\Pi}_3^S$ and the variations of this policy space that are obtained by the replacement of one of the four static random switches with the partially disaggregated scheme that was described above. In all these runs, the starting solution was the totally random policy that was described in the previous sections of this chapter, and the algorithm was executed for 500 iterations.[20]

The obtained results are summarized in Figure 5.7. This figure reveals that the

[20]In other words, the presented experiment did not employ the terminating mechanism that was presented in Section 5.3.3; this choice was made in an effort to decouple the obtained results from any potential sensitivities that might be present in the statistical tests that are employed by that mechanism.

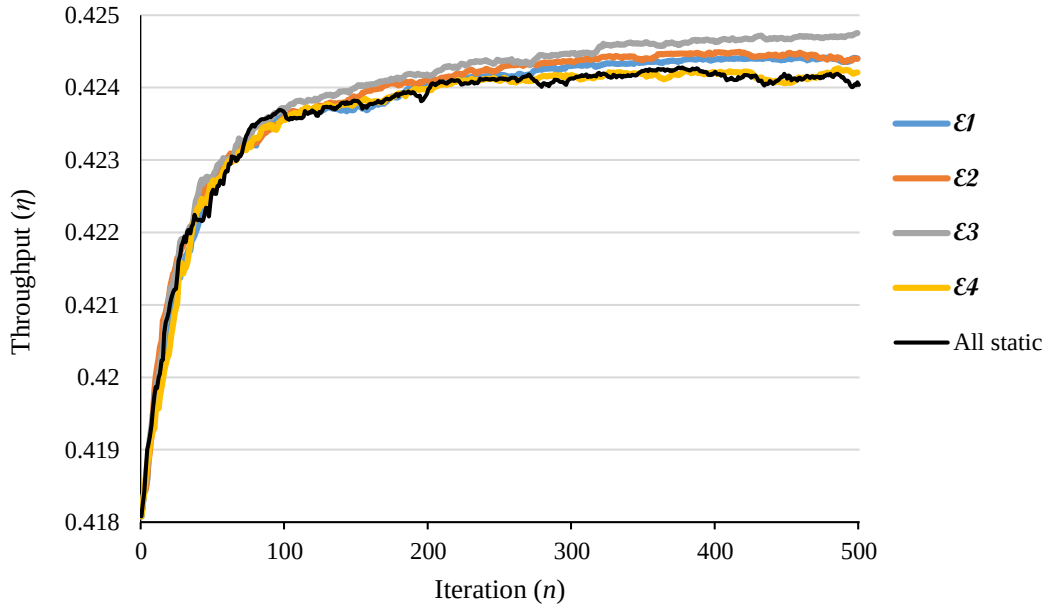


Figure 5.7: The performance improvements attained for the CRL configuration that is considered in the performed experiment, through the partial disaggregation of the static random switches of the original policy space $\hat{\Pi}_3^S$

partial disaggregation of the enabling pattern $\mathcal{E}_3$ leads to the highest performance improvement; the partial disaggregation of $\mathcal{E}_1$ and $\mathcal{E}_2$ gives less improvement than the disaggregation of $\mathcal{E}_3$; and the partial disaggregation of $\mathcal{E}_4$ does not incur any significant performance improvement with respect to the original policy space $\hat{\Pi}_3^S$.

A plausible explanation of these results can be based on the structure of the optimal policy $\bar{\zeta}^*$ that was obtained for the initial formulation on the policy space $\hat{\Pi}_3^S$. This structure can be described as follows:

$$\begin{aligned}
&\mathbb{P}\{t_0 \text{ is fired} | t_0 \text{ and } t_6 \text{ are enabled}\} = 0.058883 \\
&\mathbb{P}\{t_3 \text{ is fired} | t_3 \text{ and } t_{12} \text{ are enabled}\} = 0.065869 \\
&\mathbb{P}\{t_0 \text{ is fired} | t_0 \text{ and } t_5 \text{ are enabled}\} = 0.410361 \\
&\mathbb{P}\{t_2 \text{ is fired} | t_2 \text{ and } t_{11} \text{ are enabled}\} = 0.948579
\end{aligned} \tag{97}$$

It can be seen in the above listing that the random switches corresponding to the enabling pattern $\mathcal{E}_3$ involves the highest level of randomization among the four

random switches that are specified by $\bar{\zeta}^*$. But under the coupling logic of a static random switch, a substantial level of randomization can be an indication of a need to "compromise" antithetic trends at different markings with respect to the underlying enabling pattern, that might appear in the selection logic of the unconstrained optimal policy of the policy space $\hat{\Pi}_2$. In this case, the partial relaxation introduced in this section can enable a differentiated selection logic at different markings, and thus, it can remove the corresponding stress and enhance the attained performance. It is also true, however, that currently the impact of the proposed disaggregation scheme upon the performance of the underlying system is not fully understood.

5.4.3 Some further observations on the experiment of Section 5.4.2

In this section we take a closer look at the optimized values that were obtained in the experiment of the previous subsection, for the partially disaggregated random switch that results from the disaggregation of the static random switch corresponding to the enabling pattern $\mathcal{E}_3 = \{t_0, t_5\}$. These values, as determined at the 500-th iteration of the SA algorithm that computed the optimized policy, are reported in Table 5.5. Our primary intention in this study is to obtain some perspective on the selection logic among the corresponding transitions t_0 and t_5 that is defined by these values.

We remind the reader that in the implementation of the partial disaggregation scheme that was employed by the considered experiment, the "feature" functions for every vanishing marking M of the underlying GSPN model were the sub-markings of the process places of this GSPN model. Hence, in Table 5.5, the decision variables for the partial disaggregation of the static random switch that was defined by the enabling pattern $\mathcal{E}_3 = \{t_0, t_5\}$, have the values that are reported in the two right columns of this matrix, with each column reporting the values of the coefficients that determine the selection probability of the corresponding transition $t_i,\ i = 0, 5$ (c.f., Equation (91).) On the other hand, the first two columns of Table 5.5 report the

Table 5.5: The values at the 500-th SA iteration of the decision variables employed by the partially disaggregated random switch of $\mathcal{E}_3$

WS	JS	Place	t_0	t_5
1	1	p_0	0.00000	0.00000
		p_1	-0.07845	0.07845
2	2	p_2	-0.13720	0.13720
		p_3	-0.04658	0.04658
		p_4	-0.00432	0.00432
1	3	p_5	-0.10993	0.10993
		p_6	0.00000	0.00000
		p_7	0.20000	-0.20000
3	4	p_8	0.06181	-0.06181
		p_9	0.04317	-0.04317
		p_{10}	-0.01528	0.01528
2	5	p_{11}	-0.01777	0.01777
		p_{12}	-0.00141	0.00141

indices of the workstations (WS) and the job stages (JS) of the corresponding GSPN places.

The first thing we notice in the values that are reported in Table 5.5, is that the decision variables reported in each row of this table have opposite values. This phenomenon can be explained by the fact that the disaggregated static random switch involves only two untimed transitions.

To establish the above result for the general case, let us consider the enabling pattern $\mathcal{E} \equiv \{t_x, t_y\}$, and let p_x and p_y denote, respectively, the selection probabilities of the transitions t_x and t_y at some vanishing marking $\hat{M}$ with $\mathcal{E}_u^{\hat{\Pi}_2}(\hat{M}) = \mathcal{E}$. Furthermore, for a certain feature $\phi_{\hat{k}}(\hat{M})$, we denote by x and y the decision variables that multiply the feature function $\phi_{\hat{k}}(\cdot)$ in the respective determination of the selection probabilities p_x and p_y. Then, referring back to Equation (96), we can derive the partial derivative of p_x with respect to the decision variable x, as follows:

$$\begin{aligned} \frac{\partial}{\partial x} p_x &= p_x \cdot \psi_{\hat{k}}(\hat{M}) \cdot \left(1 - \frac{A(\hat{M}, t_x)}{B(\hat{M})}\right) \\ &= p_x \cdot \psi_{\hat{k}}(\hat{M}) \cdot (1 - p_x) \end{aligned} \tag{98}$$

Similarly, referring back to Equation (95), we can have the partial derivative of p_x with respect to y:

$$\begin{aligned}\frac{\partial}{\partial y}p_x &= p_x \cdot \left(\frac{-\psi_{\hat{k}}(\hat{M})A(\hat{M},t_y)}{B(\hat{M})}\right) \\ &= -p_x \cdot \psi_{\hat{k}}(\hat{M}) \cdot \mathbb{P}\{t_y \text{ is fired}|\hat{M}\} \\ &= -p_x \cdot \psi_{\hat{k}}(\hat{M}) \cdot (1-p_x) \end{aligned} \tag{99}$$

Comparing Equations (98) and (99), we can see that the partial derivatives $\frac{\partial}{\partial x}p_x$ and $\frac{\partial}{\partial y}p_x$ have opposite values. Therefore, if all the decision variables are initialized to zero (as is the case in the considered experiment), we will always obtain pairs of opposite numbers for the decision variables corresponding to the disaggregated random switches, at each iteration of the SA algorithm.

The next interesting observation in Table 5.5 concerns the zero values for the decision variables corresponding to the places p_0 and p_6. Since we are considering the partially disaggregated random switches with enabling pattern $\mathcal{E}_3 = \{t_0, t_5\}$, any vanishing markings M associated with this set of random switches must have the untimed transition t_0 enabled; i.e., the system at these states should be able to load a new job. Therefore, the server of workstation 1 must be idle, which implies that, in the underlying GSPN model, there must be no tokens at places p_0 and p_6; i.e., $M[p_0] = M[p_6] = 0$. The observed results then follow from the above Equations (98) and (99), and the employment of these two marking values as feature functions.

Next, we shall attempt to explain the overall pricing of the decision variables that is reported in Table 5.5, as the source of a near-optimal policy with respect to the pursued objective of throughput maximization. With this objective in the mind, intuitively, we would like (i) to maintain a sufficient number of jobs in the system to avoid the excessive starvation of the workstation servers, and also (ii) to see jobs leaving the system as quickly as possible.

To understand how the values reported in Table 5.5 serve these two intentions,

first we notice that according to the definition of the selection probabilities of partially disaggregated random switches in Equation (91), and the employed feature functions in the experiment of Section 5.4.2, a positive value of a decision variable corresponding to an untimed transition $\hat{t}$ and a place $\hat{p}$ implies an inclination of selecting $\hat{t}$ at a marking with more tokens at the place $\hat{p}$. Hence, the results of Table 5.5 imply that the probability of selecting transition t_5 over t_0 increases along with an increase of the numbers of tokens in any of the places $p_1 - p_5$ and $p_{10} - p_{12}$. Recalling that the firing of t_0 models the loading of a new job to the system, and the firing of t_5 models the advancement of an already initiated job from stage 2 to stage 3, we see that the underlying scheduling policy overall has a "last-buffer-first-served" tendency when it comes to the resolution of this particular random switch; i.e., it tends to serve objective (ii) above. At the same time, the re-entrance patterns of the considered CRL indicate that the aforementioned decision serves well objective (i), as well, especially with respect to the server of workstation 1, which is one of the "nominal" bottlenecks[21] of this line.

But there are exceptions to the aforementioned "general" trend: places p_0, $p_6 - p_9$. The reason of the zeros corresponding to places p_0 and p_6 has already been explained. To understand the deviation of the decision pattern that was discussed in the previous paragraph when it comes to places p_7, p_8 and p_9, let us first recall the meanings of these places in the original CRL configuration #13: Place p_7 models a job status where the processing has been finished at stage 3 but the job waits to be transferred to workstation 3 for its 4th processing stage. On the other hand, places p_8 and p_9 model the job status of waiting to be processed and being processed, respectively, at workstation 3 for processing stage 4. Therefore, the tokens at these 3 places model jobs that either seek the allocation of the single buffer slot at workstation 3, or they

[21] In this discussion, by "nominal" bottlenecks we imply the estimated bottlenecks when not accounting for the blocking effects that take place in the system.

have been allocated this buffer slot for the execution of processing stage 4. But then, when any of these places has a high value, a job advancement from stage 2 to stage 3 cannot really help with the depletion of existing jobs from the system, since, in this case, all jobs executing processing stages 1 to 3 must wait upon the jobs in the places p_7, p_8 and p_9; in fact, under these particular circumstances, a job advancement corresponding to transition t_5 will simply create further cluttering in workstation 1. So, in this case, transition t_0 is more preferable than transition t_5.

We conclude this section by noticing that the investigation that was presented in the above paragraphs is of a more intuitive nature, and, of course, it concerns only the (near-)optimal policy that is defined by the partial disaggregation of the static random switch that was singled out in the relevant experiment of Section 5.4.2. A more profound understanding of the structure of the optimal scheduling policy for the CRL model that is considered in this thesis, under the partial disaggregation schemes that were discussed in this section, as well as the design of a complete algorithm for the management of the overall disaggregation process, are part of the future work that is discussed in the next chapter.

CHAPTER VI

A SUMMARY OF THE MAJOR CONTRIBUTIONS AND POSSIBLE FUTURE EXTENSIONS

6.1 The major contributions

In the previous chapters of this work we have provided a complete, integrated framework for solving the performance optimization problems that arise in the operational context of many contemporary complex resource allocation systems. In particular, this framework is based on the effective utilization and the integration of the representational and computational frameworks depicted in Figure 1.2 of Chapter 1, according to the key workflow and the dependencies that are indicated in that figure.

Hence, in Chapter 2, our work has drawn from the DES supervisory control theory, and its specialization and extension for the problem of deadlock avoidance in resource allocation systems, in order to establish a correct behavior for the underlying resource allocation function. Furthermore, the corresponding results were eventually integrated in a time-based representation of the RAS operation that relies upon the modeling framework of the generalized stochastic Petri nets.

The definitive and concise semantics of the GSPN modeling framework subsequently enabled a succinct characterization of the considered performance control problem, in Chapter 3. But this characterization possesses a super-polynomial representational and computational complexity with respect to the underlying RAS structure, a manifestation of the "curse of dimensionality" that has haunted most scheduling problems of the type that is addressed in this work. In more practical terms, the negative complexity results mentioned above render intractable the effective resolution of the considered optimization problem for most RAS instantiations.

Acknowledging these complications, and keeping in line with the emerging theory of Approximate Dynamic Programming, in Chapter 4 we resorted to approximating schemes for the original formulation, focusing in particular upon a scheme that is known as "approximation in the policy space". This approximation scheme enables an explicit specification of the employed policy spaces, and a more direct control of the parameterization and the representational complexity of the candidate policies. In the application context of the RAS scheduling that is considered in this work, the specification of the target policy spaces was further facilitated by the formal representation of the RAS dynamics by the GSPN modeling framework. This representation has also enabled a systematic characterization of the sources and the nature of the suboptimality that is incurred by the problem restriction to the employed policy spaces, and it has provided effective mechanisms for establishing a systematic and controllable trade-off between the representational and computational complexity of the pursued solutions and their operational efficiency.

The solution of the resultant (approximating) formulations of the considered optimization problem was addressed in Chapter 5. The corresponding developments have sought the pertinent adaptation to the considered problem of some stochastic approximation algorithms. To this end, the adopted GSPN representation has provided a succinct and efficient simulation platform, and it has facilitated the systematic estimation of the necessary gradients. At the same time, the adopted SA algorithms have been strengthened by the integration in their basic evaluation and exploration logic of results coming from the area of statistical inference. These results have enabled our SA algorithms to proceed to a near-optimal region in a robust and stable way, while avoiding the expenditure of computational resources in areas of the underlying response surface with little potential gain.

From a more practical standpoint, the developments that are presented in this thesis enable the realization of the DES-based real-time control framework that is

depicted in Figure 1.1, and thus contributes substantially the corresponding theory of the real-time management of sequential resource allocation systems. As remarked in the opening chapter of this work, the effective control of these RAS structures is tantamount to the control of stochastic networks with extensive blocking and deadlocking effects, an area with very limited results in operations research and stochastic control theory.

Finally, the presented developments have been further customized for the particular operational context of the capacitated re-entrant line, a workflow model that has drawn extensive attention in the recent years in, both, the theoretical investigations of production scheduling and the industrial practice. This customization has also revealed the adaptability of the more general theory to the special structure and dynamics that are defined by various RAS sub-classes.

6.2 Possible future extensions

Being a first attempt to provide a complete solution to the aforementioned RAS performance optimization problems, this work has also identified a host of additional research problems that pertain to the further extension and the strengthening of the methodological base that is defined herein. Most of these problems have already been discussed in the earlier chapters of this document, but they are reviewed, organized and further expanded in this section for the benefit of the reader.

A first possible extension of this work can be along the direction of developing a more complete disaggregation algorithm based on the structure of the optimal policies obtained for the policy space $\hat{\Pi}_3^S$. This work has provided only some insights on the proposed disaggregation schemes by means of some examples, while the corresponding numerical study that is performed in Section 5.4.2 illustrates the efficacy and the potential performance improvements that can be obtained from such disaggregation. However, these developments are far from a complete partial disaggregation

scheme. To fully materialize such a scheme, we need an efficient algorithm able to automatically identify the static random switches that constitute the best options for disaggregation by providing the highest potential for performance improvement. The algorithm must also integrate additional logic that quantifies the sought trade-off between the operational efficiency of the derived policies and the computational and representational complexity involved; this capability will be especially useful for implementing a termination condition. In regards to the aforementioned trade-off between the complexity and the operational efficiency of the derived policies, we also need a more systematic approach for selecting the feature functions to be employed in the representational scheme of Section 4.3.

Another open problem that relates to the aforementioned partial disaggregation problem, but can have much broader ramifications for the proposed methodological framework, concerns the further investigation of the structure of the response surface of the original performance optimization problem and the impact upon this structure of the various aggregation schemes that are defined by the policy spaces introduced in this work. A similar issue is the better understanding of the (relative) value function of the corresponding MDP problems. Besides providing a clearer understanding of the quality and the trade-offs that are established by the pursued approximations, the availability of the aforementioned information can be effectively exploited towards the application of potentially more efficient SA algorithms in the solution of the optimization problems that were formulated in this work. As a more specific example of this possibility, we mention the actor-critic algorithm that is presented in [51]; this algorithm is considered as a pretty competitive algorithm within the broader class of SA algorithms in terms of its robustness and expediency of convergence to an optimal solution, but this potential is contingent upon the availability of a pertinent (approximating) representation of the underlying value function by a set of "feature" functions that complements a functional base that is employed by the original definition of the

algorithm.

Another interesting direction for solving the performance optimization problem of stochastic networks is through the continuous approximation of the discrete event system with a fluid model. In this set of methods, the scheduling decisions are made based on the structure of the optimal flows that are derived for the fluid model. Examples of such kind of approaches can be found in [42, 14]. With the integration of the optimal control theory, there will be a rich set of well-developed approaches for the optimization problem. In particular, the fluid-based scheduling approach of [14], that integrates elements from the theory of robust optimization, seems to be quite competitive, but it needs further adaptation to the RAS scheduling problems considered in this document.

From a more practical standpoint, the current work can be extended by applying the derived methods to RAS classes more general than the capacitated re-entrant lines that were used in this research. Two particular extensions along this line concern the application of the derived methodology on (i) RAS with more than one process type, and (ii) RAS with non-exponential processing times. As discussed in the previous parts of this document, both of these elements can be addressed by the proposed methods quite straightforwardly, but some further experimentation along these lines will strengthen the underlying implementational details, help understand better the nature of optimality in the considered policy spaces, and further demonstrate the practical potential of the methodology.

Also, in the case of performance objectives like the throughput maximization of RAS with many process types, it will be necessary to introduce additional "fairness" constraints that will contribute to the well-posedness of the corresponding optimization problem. These "fairness" constraints are essentially additional behavioral constraints that must be integrated in the structure of the RAS-modeling GSPN and be accounted for in the design of the corresponding deadlock avoidance policies.

Finally, the results developed in this work, and their further expansion along the lines that are suggested above, must be further assessed for their scalability potential, and be tested and "showcased" on some pertinent case studies drawn from the relevant industrial practice.

APPENDIX A

A MARKOV DECISION PROCESS FOR THE CONSIDERED PERFORMANCE OPTIMIZATION PROBLEM

At least in principle, the theory of Markov decision processes (MDPs) can also be used for the RAS performance optimization problem considered in this document. More specifically, the RAS performance optimization problem can be first modeled as a GSPN performance optimization problem using the method in Section 3.1, including the application of, both, a correct DAP through the one-step look-ahead scheme and the constraint that imposes the absence of deliberate idleness.[1] Then, the state space of the GSPN can be enumerated, for both the vanishing and tangible markings. Based on the enumerated state space, a discrete-time communicating MDP will be built, with the objective of maximizing the steady-state average reward. Classical solution methods for this problem can be found in [82], as long as the state space is tractable.[2] This appendix will focus on the construction of the communicating MDP.

The classical discrete-time MDP dynamics can be summarized in the following steps:

1. At period t, the system state is s_t, and there is a set of available actions $\mathcal{A}(s_t)$.

2. A decision is made to select an action $a_t \in \mathcal{A}(s_t)$ with probability $p(a_t|s_t)$.

[1] Similar to the case of the solution method through the MP formulation of Section 3.3, the deliberate idleness is prohibited by the GSPN semantics, but the sub-optimality that might be caused by this prohibition can be addressed with the method in Section 3.5.

[2] According to Section 9.5 of [82], an average reward MDP is communicating if the underlying state space of the AR-MDP is strongly connected, i.e., there exist policies under which each state is accessible from each other state. Such a property makes the classical unichain methods applicable and can give optimal solutions.

3. The system evolves to period $t+1$ and the system state becomes s_{t+1} with probability $p(s_{t+1}|s_t, a_t)$. Meanwhile, an immediate reward of $r(s_t, a_t, s_{t+1})$ is realized.

The (infinite-horizon) average reward problem defines the objective as the limit (if it exists)

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} r(s_t, a_t, s_{t+1})$$

The temporary "state" that results from the second step above, i.e., the state where the action a_t has been selected at the system state s_t, can be called a *post-decision* state [81]. In contrast, the original system states are called *pre-decision* states.

The process dynamics at a post-decision state are not impacted by any policy information. This characteristic is the same as in the case of the tangible markings of the RAS-modeling GSPNs. Therefore, in the considered MDP, all these tangible markings can be perceived as post-decision states. And the uniformization at the end of Section 3.3 can be applied on the tangible markings, to get the corresponding transition probabilities. In the considered optimization problem, the immediate rewards are defined by the uniformized reward rates of the tangible markings, and there is no immediate reward related to pre-decision states.

Next, we will specify the pre-decision states, the set of available actions for each pre-decision state, and the transition probabilities from post-decision states to pre-decision states. One natural consideration for specifying the pre-decision states are the vanishing markings. But this natural consideration is not the best way and will not be adopted in this document. Instead, the pre-decision states are defined as the markings that can be reached from post-decision states through a Markovian transition.[3] This Markovian transition can be either the firing of a timed transition in

[3]The term "Markovian transition" refers to a transition in a Markov chain. This concept is different from a "transition" in the Petri-net related concepts.

the GSPN, or a self-loop transition which is introduced by the uniformization process and leaves the discretized Markov chain at the same state. Furthermore, the pre-decision states are distinguished from each other only by their Π_2-untimed tangible reaches.[4] In other words, if two markings have the same Π_2-untimed tangible reach, then they are considered as the same pre-decision state. We must also notice at this point that the thus specified set of pre-decision states contains both vanishing and tangible markings. In particular, the treatment of tangible markings as pre-decision states is further necessitated by the dynamics that are generated by the self-loop transitions that are introduced by the uniformization process.

As a result of the previous definition of the pre-decision states, the sets of available actions at the various pre-decision states are exactly the Π_2-untimed tangible reaches, from which a tangible marking can be selected as the action taken, and thus, the next post-decision state can be determined. On the other hand, the conditional transition probabilities for any chosen action can be computed from the unformized transitional dynamics of the corresponding GSPN tangible marking. In the revisit of Example 1 below, we provide a detailed illustration of the construction of (i) the pre-decision states and (ii) the transition probabilities from post-decision states to pre-decision states.

Finally, due to the application of the relevant DAP, there exist policies establishing the accessibility of each pre-decision state from any other pre-decision state. Thus, the MDP is communicating.

Example 1 revisited In this example, we will show in detail the procedures for building all the necessary components that can define a complete communicating MDP for the performance optimization problem considered in this work, from the

[4]The application of constraint (6) of the policy space Π_3 is not necessary, since the communicating property is sufficient to define an effective exploration mechanism like those used in any MDP method, and thus, no randomization factor δ is needed.

Π_2-conditional STD of the corresponding GSPN. All the reachability-related concepts to be mentioned next are conditional with respect to the deadlock-free and non-deliberately-idling policy space Π_2.

Consider the Π_2-conditional STD presented in Figure A.1, which has already been introduced at the end of Section 3.3. The 19 tangible markings depicted in the double-circled nodes correspond to the post-decision states for the MDP depicted in the STD of Figure A.2. For instance, the tangible marking M_7 in the highlighted region in Figure A.1 corresponds to the post-decision state s_7 in Figure A.2. In the sequel, each marking in Figure A.1 will be denoted by M_i, where i is the number on the node corresponding to the marking. Also, similar notations apply in Figure A.2, with notation s_j denoting any post-decision or pre-decision state corresponding to the node labeled with j.

As already mentioned, while the immediate rewards for the pre-decision states are zero, the immediate rewards for the post-decision states are exactly the reward rates of their corresponding tangible markings divided by the uniformization rate r_{u} that was computed for this example RAS in Section 3.3. Particularly, the immediate rewards of the 8 post-decision states that are depicted in boldfaced lines in Figure A.2, are all equal to μ_3/r_{u}, since at those tangible markings, the timed transition t_7 is enabled and the corresponding reward rate is equal to μ_3. The immediate rewards of all other post-decision states are zero.

Next, we will construct the pre-decision states. As already explained, the pre-decision states are the states resulting either from the firing of a timed transition in the GSPN or from a uniformization induced self-loop transition. Obviously, the second type of pre-decision states contains all the tangible markings. In this example, each of the 19 post-decision states can be duplicated as a pre-decision state with the corresponding post-decision state as the only available action, and the transition probability from the post-decision state is exactly the self-loop transition probability.

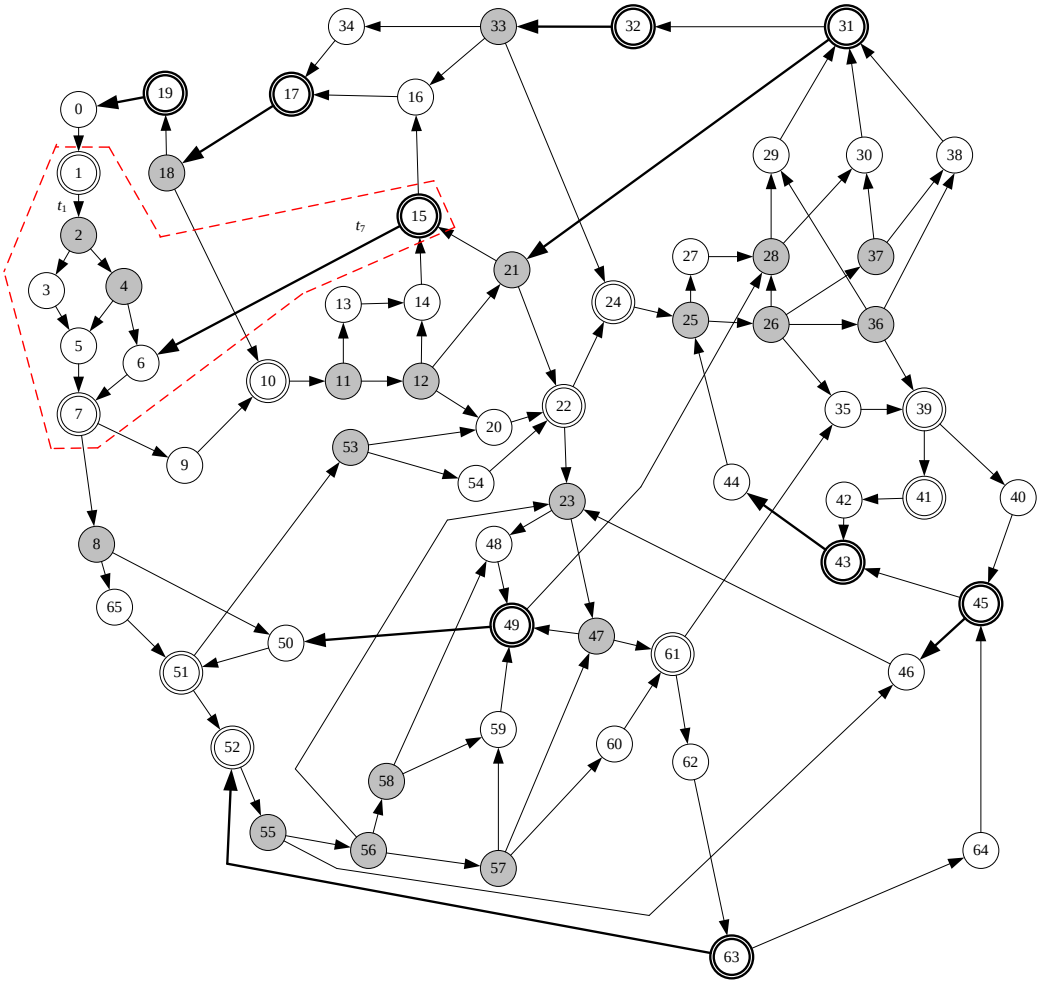


Figure A.1: The Π_2-conditional state transition diagram of the underlying semi-Markov process for Example 1 with a highlighted region on the local STD related to the construction of the example pre-decision state $s_{2,6,7'}$

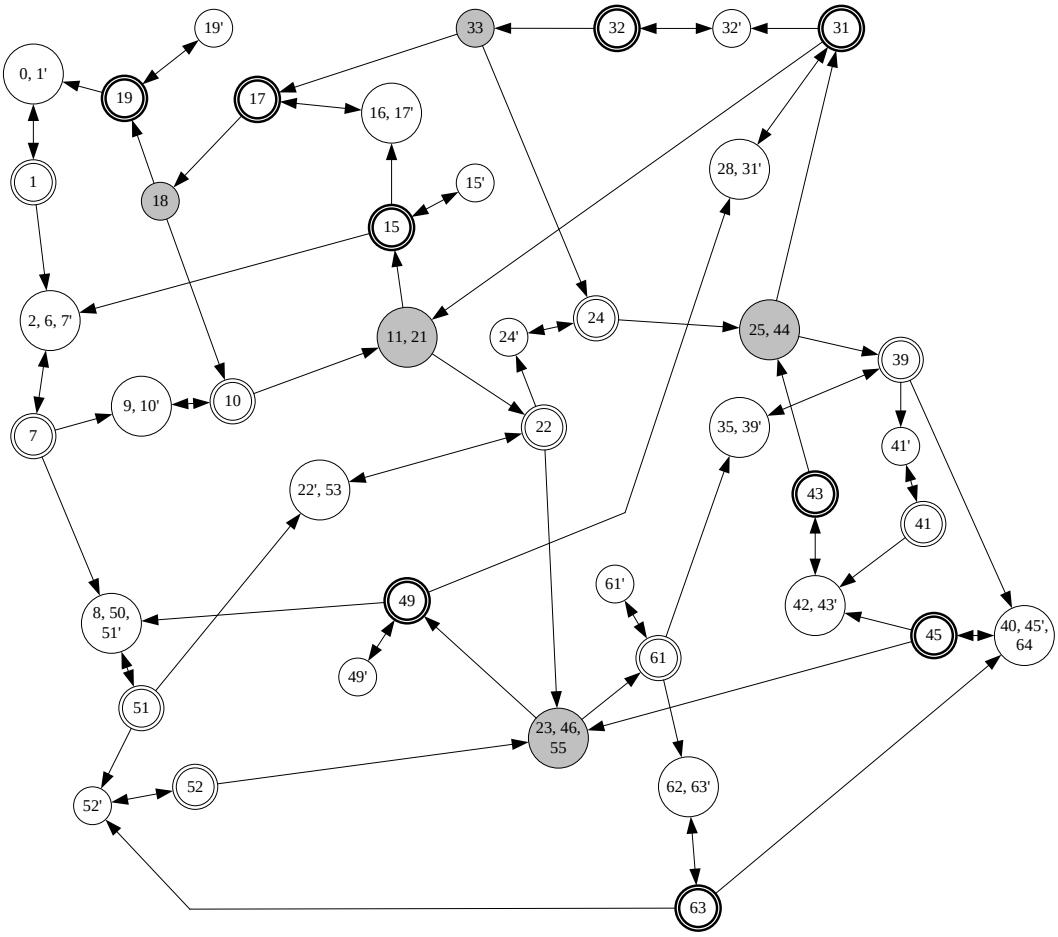


Figure A.2: The Π_2-conditional state transition diagram of the Markov decision process for Example 1

For instance, the tangible marking M_7 in the highlighted region of Figure A.1 is modeled as a pre-decision state called $s_{7'}$ in the MDP, with its set of available actions $\mathcal{A}(s_{7'}) = \{s_7\}$. Also, the corresponding transition probability from the post-decision state 7 is $(1 - \frac{\mu_1+\mu_2}{r_\mathrm{u}})$, where μ_1 and μ_2 are respectively the firing rates of timed transitions t_1 and t_4 that constitute the set $\mathcal{E}_t(M_7)$.

The other type of pre-decision states collects all the markings (either vanishing or tangible) that result immediately from the firing of timed transitions. The sets of available actions for these pre-decision states are the corresponding untimed tangible reaches, and the corresponding transition probabilities from post-decision states are determined by the rates of the corresponding timed transitions and the uniformization rate. For instance, the vanishing marking M_2 in the highlighted region of Figure A.1 is the result of firing the timed transition t_1 at the tangible marking M_1 in Figure A.1 and it is modeled as the pre-decision state s_2 in the MDP. The transition probability from post-decision state s_1 to pre-decision state s_2 is $\frac{\mu_1}{r_\mathrm{u}}$, characterized by the firing rate μ_1 of transition t_2 and the uniformization rate r_u. The Π_2-untimed tangible reach $\mathcal{UR}_{\mathcal{T}}^{\Pi_2}(M_2) = \{M_7\}$, which implies that the set of available actions $\mathcal{A}(s_2) = \{s_7\}$, and thus, is equal to $\mathcal{A}(s_{7'})$. Similar concepts apply to vanishing marking M_6 in the highlighted region of Figure A.1. It results from the firing of timed transition t_7 at tangible marking M_{15}, so the relevant transition probability is $\frac{r(t_7)}{r_\mathrm{u}} = \frac{\mu_3}{r_\mathrm{u}}$, and M_6 can be modeled as the pre-decision state s_6. Furthermore, $\mathcal{A}(s_6) = \{s_7\} = \mathcal{A}(s_{7'}) = \mathcal{A}(s_2)$.

As we have discussed before, in the constructed MDP, all the pre-decision states with the same set of available actions (or the untimed tangible reach) will be represented by a single pre-decision state. Therefore, the "single-marking" pre-decision states obtained from the above procedures can be aggregated according to their sets of available actions. For instance, the pre-decision states $s_{7'}$, s_2 and s_6 defined in the previous paragraphs have the same singleton set of available actions, and thus,

they are aggregated into one state $s_{2,6,7'}$, as depicted in Figure A.2. The transition probabilities from the post-decision states to the pre-decision state $s_{2,6,7'}$ should also be aggregated. As a result, the transition probabilities from the post-decision states s_1, s_7 and s_{15} to $s_{2,6,7'}$ are $\frac{\mu_1}{r_u}$, $(1 - \frac{\mu_1+\mu_2}{r_u})$ and $\frac{\mu_3}{r_u}$, respectively, while the transition probabilities from other post-decision states to $s_{2,6,7'}$ are zero.

In this example, after the application of the aforementioned aggregation process to all the single-marking pre-decision states, there will be 24 pre-decision states. These pre-decision states are depicted by single-circle nodes in Figure A.2. Among them, there are 5 pre-decision states with non-singleton sets of available actions. These states are depicted in gray, and they can be perceived as the "decision points". Each of these decision points has two options, or "one degree of freedom". Therefore, the total degrees of freedom for this example CRL scheduling problem is 5.

Remark The total degrees of freedom five is equal to the number of decision variables for the CRL Configuration 1 under the refinement Algorithm 2, that is reported in Table 4.2. This implies that the refinement implemented by Algorithm 2 is optimal, in the sense that it keeps only the 5 necessary decision variables corresponding to the minimum degrees of freedom specified by the MDP model.

APPENDIX B

SUPPLEMENTARY DATA

This appendix provides some supplementary data for some examples and numerical experiments that are discussed in the main part of this document.

B.1 Additional data for Example 1

Table B.1 lists the token distributions of all the Π_2-reachable markings of the GSPN in Example 1. Note that only the token distributions in process places are listed since they can uniquely define the corresponding markings.

Table B.1: Markings enumerated from the GSPN in Example 1

M	p_0p_1	$p_2p_3p_4$	p_5p_6	M	p_0p_1	$p_2p_3p_4$	p_5p_6	M	p_0p_1	$p_2p_3p_4$	p_5p_6
0	0 0	0 0 0	0 0	22	1 0	0 1 0	1 0	44	0 1	0 0 2	0 0
1	1 0	0 0 0	0 0	23	0 1	0 1 0	1 0	45	0 1	0 1 1	0 1
2	0 1	0 0 0	0 0	24	1 0	0 0 1	1 0	46	0 1	0 1 1	0 0
3	1 1	0 0 0	0 0	25	0 1	0 0 1	1 0	47	0 0	1 1 0	1 0
4	0 0	1 0 0	0 0	26	0 0	1 0 1	1 0	48	0 1	0 1 0	0 1
5	1 0	1 0 0	0 0	27	0 1	0 0 1	0 1	49	0 0	1 1 0	0 1
6	0 0	0 1 0	0 0	28	0 0	1 0 1	0 1	50	0 0	1 1 0	0 0
7	1 0	0 1 0	0 0	29	0 0	0 1 1	0 1	51	1 0	1 1 0	0 0
8	0 1	0 1 0	0 0	30	0 0	1 0 0	1 1	52	0 1	1 1 0	0 0
9	1 0	0 0 1	0 0	31	0 0	0 1 0	1 1	53	1 0	1 0 1	0 0
10	1 0	0 0 0	1 0	32	0 0	0 0 1	1 1	54	1 0	0 1 1	0 0
11	0 1	0 0 0	1 0	33	0 0	0 0 1	1 0	55	0 1	1 0 1	0 0
12	0 0	1 0 0	1 0	34	0 0	0 0 0	2 0	56	0 1	1 0 0	1 0
13	0 1	0 0 0	0 1	35	1 0	1 0 1	1 0	57	0 0	2 0 0	1 0
14	0 0	1 0 0	0 1	36	0 0	0 1 1	1 0	58	0 1	1 0 0	0 1
15	0 0	0 1 0	0 1	37	0 0	1 0 0	2 0	59	0 0	2 0 0	0 1
16	0 0	0 0 1	0 1	38	0 0	0 1 0	2 0	60	1 0	2 0 0	1 0
17	0 0	0 0 0	1 1	39	1 0	0 1 1	1 0	61	1 0	1 1 0	1 0
18	0 0	0 0 0	1 0	40	0 1	0 1 1	1 0	62	0 1	1 1 0	1 0
19	0 0	0 0 0	0 1	41	1 0	0 0 2	1 0	63	0 1	1 1 0	0 1
20	1 0	1 0 0	1 0	42	0 1	0 0 2	1 0	64	0 1	1 0 1	0 1
21	0 0	0 1 0	1 0	43	0 1	0 0 2	0 1	65	1 1	0 1 0	0 0

B.2 Detailed sample statistics for the experiment in Section 5.3.2

The tables in this section report the sample averages and the sample standard deviations of the estimates for N_{new} in the experiment of Section 5.3.2. The three tables correspond respectively to the results obtained from each of the three sample-size control methods. In each table, the sample averages ($\hat{\mu}(N_{\text{new}})$) and the sample standard deviations ($\hat{\sigma}(N_{\text{new}})$) are organized into two groups, one group for each policy under test. For each CRL configuration, the first policy, Point 1, is the totally random policy that is adopted as the initial solution of the baseline experiment in Section 5.3.1. The second policy, Point 2, is the near-optimal solution obtained through the baseline algorithm in that experiment.

Table B.2: The sample averages and standard deviations for N_{new} estimates under the Shapiro-Mello control method

Conf.	Point 1 (totally random policy)		Point 2 (near-optimal policy)	
	$\hat{\mu}(N_{\text{new}})$	$\hat{\sigma}(N_{\text{new}})$	$\hat{\mu}(N_{\text{new}})$	$\hat{\sigma}(N_{\text{new}})$
1	16.90	13.13	530.57	1663.72
2	5.37	3.21	266.32	2239.62
3	367.31	1126.81	481.15	2063.96
4	247.56	497.51	1280.07	4797.46
5	2599.37	14387.60	489.29	934.16
6	8.88	16.32	771995.47	7684909.16
7	2428.66	3861.91	822.58	4031.88
8	271.54	730.25	630.76	3648.36
9	442.58	1319.30	465.48	1419.58
10	102.82	48.79	184.85	122.78
11	55.49	59.65	202.20	718.84
12	60.62	65.79	166.55	168.52
13	33.71	25.39	243.90	231.56
14	27.34	19.56	139.89	169.91
15	116.41	134.59	115.46	75.18
16	82.80	61.83	146.36	98.48

Table B.3: The sample averages and standard deviations for N_{new} estimates under the control method that restricts the volume of the confidence region

Conf.	Point 1 (totally random policy)		Point 2 (near-optimal policy)	
	$\hat{\mu}(N_{\text{new}})$	$\hat{\sigma}(N_{\text{new}})$	$\hat{\mu}(N_{\text{new}})$	$\hat{\sigma}(N_{\text{new}})$
1	26.88	4.06	114.55	21.63
2	9.56	1.79	104.19	23.55
3	102.37	17.90	286.91	50.32
4	45.72	6.64	55.54	9.08
5	110.51	20.40	49.15	7.76
6	19.90	4.03	122.83	30.56
7	105.05	24.82	282.93	50.89
8	137.54	28.64	394.34	81.69
9	219.27	61.26	177.78	45.90
10	11.95	1.32	22.06	2.15
11	23.06	2.55	28.84	3.12
12	25.06	3.12	54.47	5.70
13	14.33	1.42	35.84	3.69
14	21.25	2.87	123.03	15.76
15	16.68	1.90	21.90	2.46
16	7.72	0.78	12.89	1.16

Table B.4: The sample averages and standard deviations for N_{new} estimates under the control method that restricts the maximum length of the confidence intervals

Conf.	Point 1 (totally random policy)		Point 2 (near-optimal policy)	
	$\hat{\mu}(N_{\text{new}})$	$\hat{\sigma}(N_{\text{new}})$	$\hat{\mu}(N_{\text{new}})$	$\hat{\sigma}(N_{\text{new}})$
1	44.16	8.39	147.12	35.58
2	10.92	2.19	105.80	23.55
3	188.76	48.40	389.48	94.03
4	68.44	14.28	101.56	24.14
5	226.12	55.40	115.08	28.77
6	21.40	3.99	124.40	30.61
7	136.64	34.78	431.24	109.03
8	195.56	59.98	529.84	164.04
9	344.84	130.60	349.80	121.05
10	54.28	15.37	292.12	72.01
11	46.84	7.51	65.16	13.43
12	54.44	10.86	172.72	30.01
13	30.32	4.73	125.24	23.08
14	36.84	6.26	315.68	68.55
15	36.72	7.33	74.84	17.24
16	17.32	2.55	58.72	10.66

APPENDIX C

AN EXAMPLE FOR THE VIOLATION OF THE Π_2-IRREDUCIBILITY OF THE REFINED TRANSITION SETS RETURNED BY ALGORITHM 3

This appendix provides an example demonstrating that the refinement attained by Proposition 8 fails to output a Π_2-irreducible subset of enabled untimed transitions with respect to the underlying marking.

The CRL considered in the example has $m = 3$ workstations and $n = 5$ processing stages. The processing route through the line workstations is $WS_1 \rightarrow WS_2 \rightarrow WS_3 \rightarrow WS_1 \rightarrow WS_2$, and the workstation buffer sizes are $(B_1, B_2, B_3) = (3, 4, 3)$ (c.f. the CRL configuration #11 of Table 4.1). In the dynamics of the GSPN modeling this CRL, the maximally permissive DAP is applied according to the one-step look-ahead method. Also, in the following discussion, the GSPN marking is represented by the sub-marking of the process places $p_0 - p_{12}$. More specifically, places p_0 and p_1 model respectively the processing and the blocking that take place at CRL stage 1; places $p_2 - p_{10}$ model the waiting, processing and blocking at the CRL stages $2 - 4$; and places p_{11} and p_{12} model respectively the waiting and processing phases at the final CRL stage 5. In the adopted representation for the aforementioned sub-marking, the place groups corresponding to distinct CRL stages are indicated by the insertion of separating spaces among them.

Next, we consider the application of Proposition 8 on the vanishing marking $M_{V1} \equiv$ (01 201 003 011 00). From the token distribution of this marking and the above description of the structure of the considered CRL, it can be inferred that

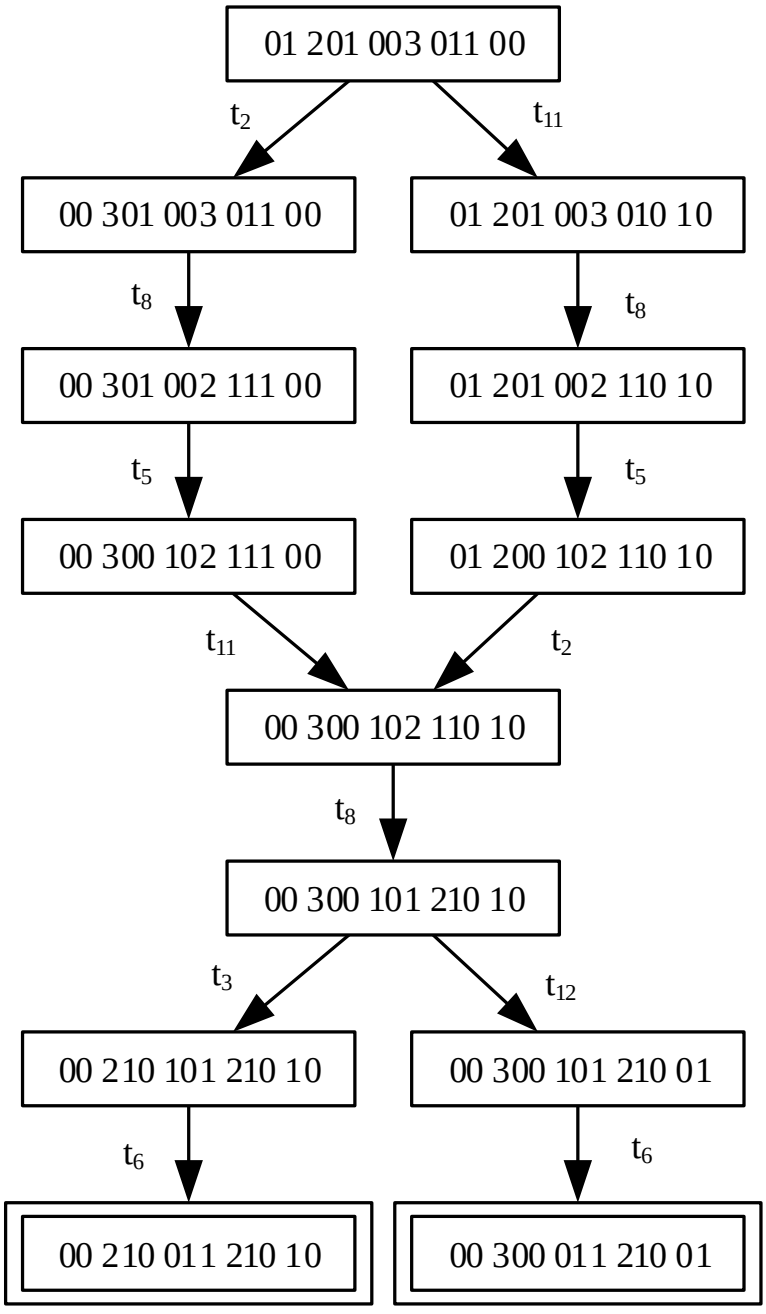


Figure C.1: The application of Propositions 7 and 8 on the untimed reach of the marking M_{V1} in the example that is considered in this appendix.

$\mathcal{E}_u^{\Pi_2}(M_{V1}) = \{t_2, t_{11}\}$, and these two transitions correspond to the respective advancement to the single free buffer slot of workstation 2 of the blocked jobs in the first and the fourth processing stage (and therefore, these two transitions are in conflict in the considered marking M_{V1}). Also, transition t_0 is not in the set $\mathcal{E}_u^{\Pi_2}(M_{V1})$. Hence, according to Proposition 8, $\mathcal{E}_u^{\hat{\Pi}_2}(M_{V1}) = \{t_2, t_{11}\}$. Let $tr(M_{V1}, t_2) = M_{V2}$ and $tr(M_{V1}, t_{11}) = M_{V3}$. It can be checked that both of these markings are vanishing, and Figure C.1 depicts the $\hat{\Pi}_2$-untimed vanishing and tangible reaches of these two last markings under the application of Propositions 7 and 8. From the figure it can be seen that $\mathcal{UR}_{\mathcal{T}}^{\hat{\Pi}_2}(M_{V2}) = \mathcal{UR}_{\mathcal{T}}^{\hat{\Pi}_2}(M_{V3})$, and thus, one of the two transitions t_2 and t_{11} can be dropped from the set $\mathcal{E}_u^{\hat{\Pi}_2}(M_{V1})$ without compromising the $\hat{\Pi}_2$-untimed tangible reach of M_{V1}.

This example reveals the interesting fact that even if two Type I transitions are in conflict in more conventional terms – i.e., the tokens in their shared input places are not enough to fire both of them – they might lead to the same behavior for the underlying CTMC when assessed with respect to the condition of Proposition 4. This apparent confusion might be perceived as a limitation of the discriminatory power of the refining logic that is defined in Proposition 8.

APPENDIX D

AN IMPLEMENTATION OF THE PROJECTION OPERATOR EMPLOYED IN THE ALGORITHMS OF CHAPTER 5

In the SA algorithms of Chapter 5, the vector $\bar{\zeta}_n + \gamma_n Y_n$ may be not in the feasible region H that is defined by Equations (34)–(35). In this case, the algorithm should find a point $\bar{\zeta}' \in H$ such that the Euclidean distance between $\bar{\zeta}'$ and $\bar{\zeta}$ is minimized. In this appendix we address the problem of computing the vector $\bar{\zeta}'$.

We start by noticing that the constraints of Equations (34)–(35) can be decomposed in constraint subsets where each subset defines the corresponding simplex for a single random switch. Hence, the projection operation that was described in the previous paragraph can be performed in a decomposing manner, by considering separately the sub-vector of $\bar{\zeta}$ that corresponds to each random switch and the corresponding simplex. And this is the approach that is discussed next.

Hence, consider a random switch R with $n+1$ options, and let the n components of the vector $\bar{\zeta}$ that correspond to this random switch be collected to another vector $x \in \mathbb{R}^n$. Furthermore, suppose that the current pricing of x is not in the simplex that is associated with this random switch, and thus, we want to find another vector $y^* \in \mathbb{R}^n$, which will replace x in $\bar{\zeta}$. The vector y^* should be an optimal solution to the following constrained nonlinear program:

$$\text{minimize} \quad \sum_{i=1}^{n}(x[i]-y[i])^2 \tag{100}$$

$$\text{subject to} \quad y[i] \geq \delta \qquad\qquad i=1,\ldots,n \tag{101}$$

$$\sum_{i=1}^{n} y[i] \leq 1-\delta \tag{102}$$

In the problem formulation (100)–(102), x and δ are the given parameters, and y is the vector of decision variables.[1]

If x itself satisfies all the constraints, then it belongs to the corresponding simplex and the above formulation has the unique optimal solution $y^* = x$. Otherwise, we have the following lemma.

Lemma 7 *If x is not in the feasible region of the optimization problem defined in* (100)–(102)*, then the violated constraints for x are binding at any optimal solution y^*.*

Proof: For the sake of contradiction, first suppose that for an index $\hat{i}$, $x[\hat{i}] < \delta$, but at an optimal solution y^*, $y^*[\hat{i}] > \delta$. Hence, another solution y' can be constructed: all the components of y' are equal to y^*, except that $y'[\hat{i}] = \delta$. Obviously y' is a feasible solution but $\sum_{i=1}^{n}(x[i]-y'[i])^2 < \sum_{i=1}^{n}(x[i]-y^*[i])^2$. This contradicts with the optimality of y^*.

Another possibility is that $\sum_{i=1}^{n} x[i] > 1-\delta$, but at an optimal solution, $\sum_{i=1}^{n} y^*[i] < 1-\delta$. So, there exists an index $\hat{i}$ such that $y^*[\hat{i}] < x[\hat{i}]$. Pick a small positive scalar ϵ such that $y^*[\hat{i}]+\epsilon \leq x[\hat{i}]$ and $\sum_{i=1}^{n} y^*[i]+\epsilon \leq 1-\delta$. As a result, another solution y' can be constructed: all the components of y' are equal to y^*, except that $y'[\hat{i}] = y^*[\hat{i}]+\epsilon$. Obviously y' is a feasible solution but $\sum_{i=1}^{n}(x[i]-y'[i])^2 < \sum_{i=1}^{n}(x[i]-y^*[i])^2$. This contradicts with the optimality of y^*. □

[1]For simplicity, the parameter δ employed in the formulation (100)–(102) is not exactly the same as the randomization factor defined in the constraints (15)–(16). However, we assume that $0 < \delta < \frac{1}{n+1}$, and therefore, the feasible region defined by (101) and (102) is non-empty.

According to Lemma 7, for all i such that $x[i] < \delta$, $y^*[i] = \delta$. Thus, the original problem formulation can be modified to:

$$\begin{aligned}
\text{minimize} \quad & \sum_{i:x[i]\geq\delta} (x[i] - y[i])^2 \\
\text{subject to} \quad & y[i] \geq \delta && \forall i, x[i] \geq \delta \\
& y[i] = \delta && \forall i, x[i] < \delta \\
& \sum_{i:x_i\geq\delta} y[i] \leq 1 - \delta(1 + |\{i : x[i] < \delta\}|)
\end{aligned}$$

If the vector x satisfies the last constraint, i.e., $\sum_{i:x_i\geq\delta} x[i] \leq 1 - \delta(1 + |\{i : x[i] < \delta\}|)$, then the optimal solution y^* is

$$y^*[i] = \begin{cases} \delta & \text{if } x[i] < \delta \\ x[i] & \text{otherwise} \end{cases}$$

Otherwise, the original optimization problem of Equations (100)–(102) can be reduced to a restricted simpler problem that solves for the values of $y^*[i]$ where $x[i] \geq \delta$. And from Lemma 7, in this new problem, the constraint (102) is binding. Without loss of generality, let us assume that the components of x are in descending order, i.e., $x[1] \geq x[2] \geq \ldots \geq x[n]$.[2] Under this assumption, there exists a unique integer m, such that for any $i > m$, $x[i] < \delta$, and for any $i \leq m$, $x[i] \geq \delta$. Then, taking into consideration all the above remarks, the original optimization problem is reduced to:

$$\text{minimize} \quad \sum_{i=1}^{m} (x[i] - y[i])^2 \tag{103}$$

$$\text{subject to} \quad y[i] \geq \delta \tag{104}$$

$$\sum_{i=1}^{m} y[i] = 1 - (n - m + 1)\delta \tag{105}$$

The objective function (103) is a strict convex function, and the feasible region is a polytope. Therefore, there exists a unique optimal solution [12]. Furthermore,

[2]In the case that the components of x are in some other arbitrary order, the problem can be solved by re-arranging the components of the optimal solution y^*.

this optimal solution must satisfy the corresponding *Karush-Kuhn-Tucker (KKT) condition* [12]. This condition stipulates that, for any optimal solution $\hat{y}$ for this problem, there exist $\lambda \in \mathbb{R}^m_{0+}, \mu \in \mathbb{R}$, such that

$$\hat{y}[i] = x[i] + \frac{\lambda[i]}{2} - \frac{\mu}{2}, \qquad i = 1, \ldots, m \tag{106}$$

$$\lambda[i] = 0 \text{ or } \hat{y}[i] = \delta, \qquad i = 1, \ldots, m \tag{107}$$

$$\sum_{i=1}^{m} \hat{y}[i] = 1 - (n - m + 1)\delta \tag{108}$$

Equation (107) implies the existence of a partition (I_1, I_2) for the index set $\{1, \ldots, m\}$ where $\lambda[i] = 0,\ \forall i \in I_1$, and $\hat{y}[i] = \delta,\ \forall i \in I_2$. Combining this partition with Equation (106), we have:

$$\begin{aligned}
\sum_{i=1}^{m} \hat{y}[i] &= \sum_{i \in I_1} \hat{y}[i] + \sum_{i \in I_2} \hat{y}[i] \\
&= \sum_{i \in I_1} (x[i] - \frac{\mu}{2}) + |I_2|\delta \\
&= \sum_{i \in I_1} x[i] - \frac{\mu |I_1|}{2} + |I_2|\delta
\end{aligned} \tag{109}$$

Matching the right-hand-sides of Equations (109) and (108), we have:

$$\sum_{i \in I_1} x[i] - \frac{\mu |I_1|}{2} + |I_2|\delta = 1 - (n - m + 1)\delta$$

Equivalently,

$$\begin{aligned}
-\frac{\mu}{2} &= \frac{1}{|I_1|}(-\sum_{i \in I_1} x[i] + 1 - (n - m + |I_2| + 1)\delta) \\
&= \frac{1}{|I_1|}(-\sum_{i \in I_1} x[i] + (1 - (n - |I_1| + 1)\delta)) \\
&= -\frac{1}{|I_1|}\sum_{i \in I_1} x[i] + \frac{1}{|I_1|}(1 - (n + 1)\delta) + \delta
\end{aligned}$$

Finally, the solution of $\hat{y}$ is

$$y^*[i] = \begin{cases} x[i] - \frac{1}{|I_1|}\sum_{j\in I_1} x[j] + \frac{1}{|I_1|}(1-(n+1)\delta) + \delta\,, & \text{if } i \in I_1 \\ \delta\,, & \text{if } i \in I_2 \end{cases} \tag{110}$$

To effectively construct the partition (I_1, I_2), we work with Equation (110) and the constraints $y^*[i] \geq \delta, i \in I_1$, and $\lambda[i] \geq 0, i \in I_2$. Therefore, for any $i \in I_1$, we stipulate that

$$x[i] - \frac{1}{|I_1|}\sum_{j\in I_1} x[j] + \frac{1}{|I_1|}(1-(n+1)\delta) + \delta \geq \delta$$

A more useful restatement of the above inequality is as follows:

$$\frac{1}{|I_1|}\sum_{j\in I_1} x[j] - \frac{1}{|I_1|}(1-(n+1)\delta)) \leq \min_{i\in I_1} x[i] \tag{111}$$

On the other hand, for any $i \in I_2$:

$$\begin{aligned} \delta &= y^*[i] \\ &= x[i] + \frac{\lambda[i]}{2} - \frac{\mu}{2} \\ &= x[i] + \frac{\lambda[i]}{2} - \frac{1}{|I_1|}\sum_{j\in I_1} x[j] + \frac{1}{|I_1|}(1-(n+1)\delta) + \delta \end{aligned}$$

Hence, for any $i \in I_2$

$$\frac{\lambda[i]}{2} = -x[i] + \frac{1}{|I_1|}\sum_{j\in I_1} x[j] - \frac{1}{|I_1|}(1-(n+1)\delta) \geq 0$$

or equivalently

$$\frac{1}{|I_1|}\sum_{j\in I_1} x[j] - \frac{1}{|I_1|}(1-(n+1)\delta) \geq \max_{i\in I_2} x[i] \tag{112}$$

Equations (111) and (112), together with the assumed decreasing ordering of the components of the vector x, imply that for any feasible partition (I_1, I_2) there must exist an index $i^* \in \{1, \ldots, m\}$ such that any $i \leq i^*$ is put into I_1 and the remaining indices are put into I_2. It is possible that $I_2 = \emptyset$, but I_1 is always non-empty since, for

any meaningful value of δ, Equation (111) will always be satisfied by picking $I_1 = \{1\}$. For a given i^*, the corresponding vector y^* is given by:

$$y^*[i] = \begin{cases} x[i] - \frac{1}{i^*}\sum_{j=1}^{i^*} x[j] + \frac{1}{i^*}(1-(n+1)\delta) + \delta\,, & \text{if } i \leq i^* \\ \delta\,, & \text{otherwise} \end{cases} \tag{113}$$

A simple algorithm to identify the index i^* is by enumerating the integers from 1 to m, and stopping at the value where the corresponding partition (I_1, I_2) satisfies both Equations (111) and (112). The existence of such a feasible i^* is guaranteed by the existence of a solution to the KKT conditions. The choice of i^* may not be unique, but the solution y^* that is specified by Equation (113) is unique.

Proposition 10 *The solution y^* for the formulation* (103)–(105) *that is specified by Equation* (113) *is unique.*

Proof: The developments of this appendix imply that every solution to the KKT conditions is given by Equation (113) for some i^* inducing a partition (I_1, I_2) that satisfies the conditions of Equations (111) and (112). Hence, suppose that any two indices i_1, i_2 $(i_1 + 1 \leq i_2)$ specify two feasible partitions with respect to Equations (111) and (112). Then, it suffices to prove that the corresponding solutions y_1^* and y_2^*, that are defined by Equation (113), are the same.

Since i_1 and i_2 specify feasible partitions, then:

$$\frac{1}{i_1}\sum_{j=1}^{i_1} x[j] - x[i_1] \leq \frac{1}{i_1}(1-(n+1)\delta)$$

$$\frac{1}{i_1}\sum_{j=1}^{i_1} x[j] - x[i_1+1] \geq \frac{1}{i_1}(1-(n+1)\delta)$$

and

$$\frac{1}{i_2}\sum_{j=1}^{i_2} x[j] - x[i_2] \leq \frac{1}{i_2}(1-(n+1)\delta)$$

$$\frac{1}{i_2}\sum_{j=1}^{i_2} x[j] - x[i_2+1] \geq \frac{1}{i_2}(1-(n+1)\delta)$$

Equivalently,

$$\begin{aligned}
\sum_{j=1}^{i_1}(x[j]-x[i_1]) &\leq 1-(n+1)\delta \\
\sum_{j=1}^{i_1}(x[j]-x[i_1+1]) &\geq 1-(n+1)\delta \\
\sum_{j=1}^{i_2}(x[j]-x[i_2]) &\leq 1-(n+1)\delta \\
\sum_{j=1}^{i_2}(x[j]-x[i_2+1]) &\geq 1-(n+1)\delta
\end{aligned} \tag{114}$$

Since the components of x are in descending order, the left-hand-sides of the inequality system (114) can be re-arranged as follows:

$$\begin{aligned}
\sum_{j=1}^{i_1}(x[j]-x[i_1]) &\leq \sum_{j=1}^{i_1}(x[j]-x[i_1+1]) \\
&\leq \sum_{j=1}^{i_1+1}(x[j]-x[i_1+1]) \\
&\leq \sum_{j=1}^{i_1+1}(x[j]-x[i_1+2]) \\
&\leq \cdots \\
&\leq \sum_{j=1}^{i_2}(x[j]-x[i_2]) \\
&\leq \sum_{j=1}^{i_2}(x[j]-x[i_2+1])
\end{aligned}$$

Combining this last set of inequalities with the inequalities provided in (114), we get that

$$
\begin{aligned}
1-(n+1)\delta &= \sum_{j=1}^{i_1}(x[j]-x[i_1+1]) \\
&= \sum_{j=1}^{i_1+1}(x[j]-x[i_1+1]) \\
&= \sum_{j=1}^{i_1+1}(x[j]-x[i_1+2]) \\
&= \ldots \\
&= \sum_{j=1}^{i_2}(x[j]-x[i_2])
\end{aligned}
\tag{115}
$$

The result follows from the combination of Equation (115) with Equation (113) that specifies the solutions y_1^* and y_2^* for the corresponding partitions that are defined by i_1 and i_2. We leave the relevant verification to the reader. $\square$

The complete projection algorithm that is defined from the above developments is formally stated as Algorithm 7.

Algorithm 7 Projection of the variable vector defining a single random switch of the formulation (33)–(35) on the corresponding simplex

Input: $GSPN^S$, Π, M, $\bar{\zeta}$, $idx(\cdot)$, the randomizing parameter δ.
Output: The projected value of $\bar{\zeta}$.

$sum \leftarrow 0$; $\delta \leftarrow \delta/|\mathcal{E}_u^{\Pi}(M)|$; $S \leftarrow \{i : \exists t \in \mathcal{E}_u^{\Pi}(M) \text{ s.t. } idx(M,t) = i\}$.
for $i \in S$ **do**
  if $\bar{\zeta}[i] < \delta$ **then**
    $\bar{\zeta}[i] \leftarrow \delta$.
  end if
  $sum \leftarrow sum + \bar{\zeta}[i]$.
end for
if $1 - sum < \delta$ **then**
  $y \leftarrow$ vector of components in $\bar{\zeta}[S]$ in descending order; define mapping $j = key(i)$ such that $y[i]$ stands for $\bar{\zeta}[j]$ for some $j \in S$.
  for $i = 1 \rightarrow |S|$ **do**
    $\mu \leftarrow \frac{1}{i}\sum_{k=1}^{i} y[k]$.
    $resid \leftarrow (1 - |\mathcal{E}_u^{\Pi}(M)|\delta)/i$.
    if $\mu - y[i] \leq resid \wedge \mu - y[i+1] \geq resid$ **then**
      Exit the for-loop.
    end if
  end for
  for $k = 1 \rightarrow i$ **do**
    $\bar{\zeta}[key(k)] \leftarrow y[k] - \mu + resid + \delta$.
  end for
  for $k = i + 1 \rightarrow |S|$ **do**
    $\bar{\zeta}[key(k)] \leftarrow \delta$.
  end for
end if
return $\bar{\zeta}$.

REFERENCES

[1] Ajmone Marsan, M., Balbo, G., Conte, G., Donatelli, S., and Franceschinis, G., *Modelling with generalized stochastic Petri nets*. Chichester, UK: John Wiley & Sons, 1995.

[2] Ajmone Marsan, M., Conte, G., and Balbo, G., "A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems," *ACM Transactions on Computer Systems*, vol. 2, no. 2, pp. 93–122, 1984.

[3] Asmussen, S., *Applied Probability and Queues (2nd ed.)*. NY, NY: Springer-Verlag, 2003.

[4] Asmussen, S. and Glynn, P. W., *Stochastic Simulation: Algorithm and Analysis*. NY, NY: Springer, 2007.

[5] Banaszak, Z. A. and Krogh, B. H., "Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 6, pp. 724–734, 1990.

[6] Barkaoui, K., Chaoui, A., and Zouari, B., "Supervisory control of discrete event systems based on structure theory of Petri nets," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 3750–3755, IEEE, 1997.

[7] Bastin, F., Cirillo, C., and Toint, P. L., "An adaptive Monte Carlo algoriothm for computing mixed logit estimators," *Computational Management Science*, vol. 3, no. 1, pp. 55–79, 2006.

[8] Bellman, R., *Applied Dynamic Programming*. Princeton, N. J.: Princeton University, 1957.

[9] Benveniste, A., Métivier, M., and Priouret, P., *Adaptive Algorithms and Stochastic Approximations*. Springer-Verlag, 1990.

[10] Bernardo, J. M. and Smith, A. F. M., *Bayesian Theory*. John Wiley & Sons, 1994.

[11] Bertsekas, D. P., *Dynamic Programming and Optimal Control, Vol. II (3th ed.)*. Nashua, NH: Athena Scientific, 2007.

[12] Bertsekas, D. P., Nedić, A., and Ozdaglar, A. E., *Convex Analysis and Optimization*. Athena Scientific, 2003.

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

[14] Bertsimas, D., Nasrabadi, E., and Paschalidis, I. C., "Robust fluid processing networks," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 715–728, 2015.

[15] Byrd, R. H., Chin, G. M., Nocedal, J., and Wu, Y., "Sample size selection in optimization methods for machine learning," *Mathematical Programming*, vol. 134, no. 1, pp. 127–155, 2012.

[16] Cassandras, C. G. and Lafortune, S., *Introduction to Discrete Event Systems (2nd ed.)*. NY,NY: Springer, 2008.

[17] Chen, P., Bruell, S. C., and Balbo, G., "Alternative methods for incorporating non-exponential distribution into stochastic timed Petri nets," in *Proceedings of the 3rd International Workshop on Petri Nets and Performance Models*, pp. 187–197, IEEE, 1989.

[18] Chiola, G., Donatelli, S., and Franceschinis, G., "GSPN versus SPN: what is the actual role of immediate transitions?," in *Proceedings of the 4th International Workshop on Petri Nets and Performance Models*, pp. 20–31, IEEE, 1991.

[19] Choi, J. Y., *Performance Modelling, Analysis and Control of Capacitated Reentrant Lines*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, 2004.

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

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

[22] Ciardo, G., Muppala, J. K., and Trivedi, K. S., "On the solution of GSPN reward models," *Performance Evaluation*, vol. 12, no. 4, pp. 237–253, 1991.

[23] Coffman, E. G., Elphick, M. J., and Shoshani, A., "System deadlocks," *Computing Surveys*, vol. 3, no. 2, pp. 67–78, 1971.

[24] Dai, J. and Weiss, G., "Stability and instability of fluid models for reentrant lines," *Mathematics of Operations Research*, vol. 21, no. 1, pp. 115–134, 1996.

[25] David, R. and Alla, H., *Discrete, Continuous, and Hybrid Petri Nets*. Berlin, Germany: Springer, 2005.

[26] Deng, G. and Ferris, M. C., "Variable-number sample-path optimization," *Mathematical Programming*, vol. 117, no. 1, pp. 81–109, 2009.

[27] Ermoliev, Y. and Wets, R. J. B., eds., *Numerical Techniques for Stochastic Optimization*. Berlin, Germany: Springer-Verlag, 1988.

[28] Estanjini, R. M., Li, K., and Paschalidis, I. C., "A least squares temporal difference actorcritic algorithm with applications to warehouse management," *Naval Research Logistics*, vol. 59, no. 3–4, pp. 197–211, 2012.

[29] Ezpeleta, J., Colom, J. M., and Martínez, J., "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 2, pp. 173–184, 1995.

[30] Ezpeleta, J., García-Vallés, F., and Colom, J. M., "A class of well structured Petri nets for flexible manufacturing systems," in *Proceedings of the 19th International Conference on Application and Theory of Petri Nets* (Desel, J. and Silva, M., eds.), pp. 64–83, 1998.

[31] Ezpeleta, J., Tricas, F., García-Vallés, F., and Colom, J. M., "A Banker's solution for deadlock avoidance in FMS with flexible routing and multiresource states," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 4, pp. 621–625, 2002.

[32] Fanti, M. P., Maione, B., Mascolo, S., and Turchiano, B., "Event-based feedback control for deadlock avoidance in flexible production systems," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 3, pp. 347–363, 1997.

[33] Fei, Z., Akesson, K., and Reveliotis, S., "Symbolic computation and representation of deadlock avoidance policies for complex resource allocation systems with application to multithreaded software," in *Proceedings of the 53rd IEEE Conference on Decision and Control*, pp. 5935–5942, IEEE, 2014.

[34] Fei, Z., Reveliotis, S., Miremadi, S., and Akesson, K., "A BDD-based approach for designing maximally permissive deadlock avoidance policies for complex resource allocation systems," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 990–1006, 2015.

[35] Fu, M. C., "Gradient estimation," in *Handbooks in Operations Research and Management Science: Simulation* (Henderson, S. G. and Nelson, B. L., eds.), pp. 575–616, Elsevier, 2006.

[36] Fu, M. C., ed., *Handbook of Simulation Optimization*. Springer, 2015.

[37] Giua, A., DiCesare, F., and Silva, M., "Generalized mutual exclusion constraints on nets with uncontrollable transitions," in *Proceedings of the 1992 IEEE International Conference on Systems, Man and Cybernetics*, pp. 974–979, IEEE, 1992.

[38] Glasserman, P. and Yao, D., *Monotone Structure in Discrete-Event Systems*. NY, NY: John Wiley & Sons, 1994.

[39] Gold, E. M., "Deadlock prediction: Easy and difficult cases," *SIAM Journal of Computing*, vol. 7, no. 3, pp. 320–336, 1978.

[40] Grötschel, M., Lovász, L., and Schrijver, A., *Geometric Algorithms and Combinatorial Optimization*. Algorithms and Combinatorics 2, Springer-Verlag, 1998.

[41] Hashemi, F. S., Ghosh, S., and Pasupathy, R., "On adaptive sampling rules for stochastic recursions," in *Proceedings of the 2014 Winter Simulation Conference*, pp. 3959–3970, IEEE, 2014.

[42] Henderson, S. G., Meyn, S. P., and Tadić, V. B., "Performance evaluation and policy selection in multiclass networks," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 13, no. 1, pp. 149–189, 2003.

[43] Ho, Y. C., Zhao, Q. C., and Jia, Q. S., *Ordinal Optimization: Soft Optimization for Hard Problems*. NY, NY: Springer, 2007.

[44] Homem-de Mello, T., *Simulation-Based Methods for Stochastic Optimization*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, 1998.

[45] Homem-de Mello, T., "Variable-sample methods for stochastic optimization," *ACM Transactions on Modeling and Computer Simulation*, vol. 13, no. 2, pp. 108–133, 2003.

[46] Hutchison, D. W. and Spall, J. C., "A method for stopping nonconvergent stochastic approximation processes," in *Proceedings of the 44th Conference on Decision and Control, and the European Control Conference 2005*, pp. 6620–6625, IEEE, 2005.

[47] Iordache, M. V. and Antsaklis, P. J., *Supervisory Control of Concurrent Systems: A Petri Net Structural Approach*. Boston, MA: Birkhäuser, 2006.

[48] Jeng, M., Xie, X., and Peng, M. Y., "Process nets with resources for manufacturing modeling and their analysis," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 6, pp. 875–889, 2002.

[49] Johnson, N. L., Kotz, S., and Balakrishnan, N., *Continuous Univariate Distributions*. John Wiley & Sons, 1994.

[50] Kindermann, R. and Snell, J. L., *Markov Random Fields and Their Applications*. Contemporary Mathematics, Providence, RI: American Mathematical Society, 1980.

[51] Konda, V., *Actor-Critic Algorithms*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2002.

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

[53] Kushner, H. J. and Yin, G. G., *Stochastic Approximation and Recursive Algorithms and Applications*. NY, NY: Springer, 2003.

[54] Lawley, M., Reveliotis, S., and Ferreira, P., "Flexible manufacturing system structural control and the Neighborhood Policy, part 1. correctness and scalability," *IIE Transactions*, vol. 29, no. 10, pp. 877–887, 1997.

[55] Lawley, M., Reveliotis, S., and Ferreira, P., "Flexible manufacturing system structural control and the Neighborhood Policy, part 2. generalization, optimization and efficiency," *IIE Transactions*, vol. 29, no. 10, pp. 889–899, 1997.

[56] Lawley, M., Reveliotis, S., and Ferreira, P., "The application and evaluation of Banker's algorithm for deadlock-free buffer space allocation in flexible manufacturing systems," *International Journal of Flexible Manufacturing Systems*, vol. 10, no. 1, pp. 73–100, 1998.

[57] Lawley, M., Reveliotis, S., and Ferreira, P., "A correct and scalable deadlock avoidance policy for flexible manufacturing systems," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 5, pp. 796–809, 1998.

[58] Lawley, M. A. and Reveliotis, S. A., "Deadlock avoidance for sequential resource allocation systems: hard and easy cases," *International Journal of Flexible Manufacturing Systems*, vol. 13, no. 4, pp. 385–404, 2001.

[59] Li, R. and Reveliotis, S., "Performance optimization for a class of generalized stochastic Petri nets," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 25, no. 3, pp. 387–417, 2015.

[60] Li, R. and Reveliotis, S., "Designing parsimonious scheduling policies for complex resource allocation systems through concurrency theory," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 26, no. 3, pp. 511–537, 2016.

[61] Li, Z. and Zhou, M., *Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach*. Springer, 2009.

[62] Li, Z., Zhou, M., and Wu, N., "A survey and comparison of Petri net-based deadlock prevention policies for flexible manufacturing systems," *IEEE Transactions on Systems, Man and Cybernetics – Part C: Applications and Reviews*, vol. 38, no. 2, pp. 173–188, 2008.

[63] Liao, H., Lafortune, S., Reveliotis, S., Y., W., and Mahlke, S., "Optimal liveness-enforcing control of a class of Petri nets arising in multithreaded software," *IEEE Transactions on Automatic Control*, vol. 58, no. 5, pp. 1123–1138, 2013.

[64] Liao, H., Wang, Y., Cho, H. K., Stanley, J., Kelly, T., Lafortune, S., Mahlke, S., and Reveliotis, S., "Concurrency bugs in multithreaded software: Modeling and analysis using Petri nets," *Discrete Event Systems: Theory and Applications*, vol. 23, no. 2, pp. 157–195, 2013.

[65] Liao, H., Wang, Y., Stanley, J., Lafortune, S., Reveliotis, S., Kelly, T., and Mahlke, S., "Eliminating concurrency bugs in multithreaded software: A new approach based-on discrete-event control," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 6, pp. 2067–2082, 2013.

[66] Lu, S. H. and Kumar, P. R., "Distributed scheduling based on due dates and buffer priorities," *IEEE Transactions on Automatic Control*, vol. 36, no. 12, pp. 1406–1416, 1991.

[67] Luenberger, D. G., *Linear and Nonlinear Programming (2nd ed.)*. Reading, MA: Addison Wesley, 1984.

[68] Marbach, P. and Tsitsiklis, J. N., "Simulation-based optimization of Markov reward processes," *IEEE Transactions on Automatic Control*, vol. 46, no. 2, pp. 191–209, 2001.

[69] Mathai, A. M. and Provost, S. B., *Quadratic Forms in Random Variables: Theory and Applications*. NY, NY: Marcel Dekker, 1992.

[70] Meyer, C. D., *Matrix Analysis and Applied Linear Algebra*. Philadelphia, PA: SIAM, 2000.

[71] Montgomery, D. C. and Runger, G. C., *Applied Statistics and Probability for Engineers (3rd ed.)*. NY, NY: John Wiley & Sons, 2003.

[72] Moody, J. O. and Antsaklis, P. J., *Supervisory Control of Discrete Event Systems using Petri Nets*. Boston, MA: Kluwer Academic Pub., 1998.

[73] Murata, T., "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[74] Nazeem, A., *Designing parsimonious representations of the maximally permissive deadlock avoidance policy for complex resource allocation systems through classification theory*. PhD thesis, Georgia Tech, Atlanta, GA, 2012.

[75] Nazeem, A. and Reveliotis, S., "A practical approach for maximally permissive liveness-enforcing supervision of complex resource allocation systems," *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 4, pp. 766–779, 2011.

[76] Nazeem, A. and Reveliotis, S., "Efficient enumeration of minimal unsafe states in complex resource allocation systems," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 1, pp. 111–124, 2014.

[77] Nazeem, A., Reveliotis, S., Wang, Y., and Lafortune, S., "Designing compact and maximally permissive deadlock avoidance policies for sequential resource allocation systems through classification theory: the linear case," *IEEE Transactions on Automatic Control*, vol. 56, no. 8, pp. 1818–1833, 2011.

[78] Pallottino, L., Scordio, V. G., Bicchi, A., and Frazzoli, E., "Decentralized cooperative policy for conflict resolution in multivehicle systems," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1170–1183, 2007.

[79] Park, J. and Reveliotis, S. A., "Deadlock avoidance in sequential resource allocation systems with multiple resource acquisitions and flexible routings," *IEEE Transactions on Automatic Control*, vol. 46, no. 10, pp. 1572–1583, 2001.

[80] Pasupathy, R. and Schmeiser, B. W., "Root finding via DARTS: Dynamic adaptive random target shooting," in *Proceedings of the 2010 Winter Simulation Conference*, pp. 1255–1262, IEEE, 2010.

[81] Powell, W. B., *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. N.Y., N.Y: Wiley, 2007.

[82] Puterman, M. L., *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ: John Wiley & Sons, 2005.

[83] Reveliotis, S. A., "Conflict resolution in AGV systems," *IIE Transactions*, vol. 32, no. 7, pp. 647–659, 2000.

[84] Reveliotis, S. A., "The destabilizing effect of blocking due to finite buffering capacity in multi-class queueing networks," *IEEE Transactions on Automatic Control*, vol. 45, no. 3, pp. 585–588, 2000.

[85] Reveliotis, S. A., "On the siphon-based characterization of liveness in sequential resource allocation systems," in *Applications and Theory of Petri Nets 2003*, pp. 241–255, 2003.

[86] Reveliotis, S. A., *Real-time Management of Resource Allocation Systems: A Discrete Event Systems Approach*. NY, NY: Springer, 2005.

[87] Reveliotis, S. A., "Coordinating autonomy: Sequential resource allocation systems for automation," *IEEE Robotics and Automation Magazine*, vol. 22, no. 2, pp. 77–94, 2015.

[88] Reveliotis, S. A., Lawley, M. A., and Ferreira, P. M., "Polynomial complexity deadlock avoidance policies for sequential resource allocation systems," *IEEE Transactions on Automatic Control*, vol. 42, no. 10, pp. 1344–1357, 1997.

[89] Reveliotis, S. A. and Nazeem, A., "Deadlock avoidance policies for automated manufacturing systems using finite state automata," in *Formal Methods in Manufacturing* (Campos, J., Seatzu, C., and Xie, X., eds.), pp. 169–195, CRC Press / Taylor & Francis, 2014.

[90] Reveliotis, S. A. and Roszkowska, E., "On the complexity of maximally permissive deadlock avoidance in multi-vehicle traffic systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 7, pp. 1646–1651, 2010.

[91] Reveliotis, S. A. and Roszkowska, E., "Conflict resolution in free-ranging multivehicle systems: A resource allocation paradigm," *IEEE Transactions on Robotics*, vol. 27, no. 2, pp. 283–296, 2011.

[92] Rosenthal, J. S., *A First Look at Rigorous Probability Theory, 2nd Ed.* World Scientific, 2006.

[93] Ross, S. M., *Introduction to Probability Models, 9th Ed.* San Diego, CA: Academic Press, Inc., 2007.

[94] Seatzu, C., Silva, M., and van Schuppen, J. H., eds., *Control of Discrete Event Systems: Automata and Petri Net Perspectives*. NY, NY: Springer, 2013.

[95] Shapiro, A. and Homem-de Mello, T., "A simulation-based approach to two-stage stochastic programming with recourse," *Mathematical Programming*, vol. 81, no. 3, pp. 301–325, 1998.

[96] Sielken, R. L., "Stopping times for stochastic approximation procedures," *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, vol. 26, no. 1, pp. 67–75, 1973.

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

[98] Spall, J. C., "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 332–341, 1992.

[99] Tricas, F., García-Vallés, F., Colom, J. M., and Ezpeleta, J., "A structural approach to the problem of deadlock prevention in processes with resources," in *Proceedings of the 4th Workshop on Discrete Event Systems*, pp. 273–278, IEE, 1998.

[100] Viswanadham, N., Narahari, Y., and Johnson, T. L., "Stochastic modelling of flexible manufacturing systems," *Mathematical and Computer Modelling*, vol. 16, no. 3, pp. 15–34, 1992.

[101] Yin, G., "A stopped stochastic approximation algorithm," *Systems & Control Letters*, vol. 11, no. 2, pp. 107–115, 1988.

[102] Zhou, M. and Fanti, M. P., *Deadlock Resolution in Computer-Integrated Systems*. Boca Raton, FL: Marcel Dekker, Inc./CRC Press, 2004.

VITA

Ran Li was born in Beijing, China. He received the Bachelor's Degree in Industrial Engineering from Tsinghua University, Beijing, China, in 2007, and the Master's Degree in Management Science and Engineering from Tsinghua University, in 2010. He was a Visiting Researcher in Rotterdam School of Management, Erasmus University, the Netherlands, in 2009. Since 2010 he has been a Ph.D. student in Industrial Engineering at Georgia Institute of Technology, GA.